```
/************************************************************
 * Version          : 3.0.5
 * Source file      : c_d_Convert.cpp
 * File summary     : Conversion processing main file
 *                  : with Calculate fuel
 * Created by:
 * Updated on (created on) : 2003.11.10(2003.11.10)
 * Remarks      : Compile switches for compiling are listed below.
 *                  :
 *  _MSC_VER    : Microsoft Visual C++ ver 6
 *  _GNUC_      : GNU C++/GCC/G++
 *          : Borland C++Builder 6
 * HISTORY      :
 * ID  -- DATE -- ----- NOTE ---------------------------------
 * 00  2003.11.10 Created
 * 01  2006.02.22 Display and File input changed.
 * 02  2006.03.23 2 point mapout and output common file recover.
 * 03  2006.04.11 AT calculate paramater recover V3.0.1
 * 04  2006.04.13 display output recover V3.0.2
 * 05  2006.04.15 shift up algo. recover V3.0.3
 * 06  2007.01.23 CCsvFile readed pattern is space-only delete V3.0.4
 * 07  2007.07.09 Add Revision calculation for TT1,T10,T11 V3.0.5
 ************************************************************/
#ifndef __CONVERT__
#define __CONVERT__
#define VERSIONNO   "3.0.5"
// ---------------------------------------------------------------------------
// Include
// ---------------------------------------------------------------------------
#include "mapout.h"
#include <math.h>

#include <stdio.h>
#include <iostream>
#include <map>
#include <set>
#include <string>
#include <vector>


//---------------------------------------
// Environmental switch
//---------------------------------------
#ifndef _MSC_VER
    #pragma package(smart_init)
    #ifndef __GNUC__
        #include <dir.h>
        #include <process.h>
        #include <mem.h>
    #endif
#else
    #include <io.h>
    #include <errno.h>
    #include <direct.h>
    #include <Memory.h>
    #include <FLOAT.h>
    #define _USE_MATH_DEFINES
    #pragma optimize("g",off)
#endif

#ifdef __GNUC__
    #define __STL_HAS_NAMESPACES
    #include <unistd.h>
    #include <conio.h>
    #include <windows.h>
#else
    #include <conio.h>
    #include <windows.h>
#endif

#pragma hdrstop
using namespace std;


//===========================================================================
// DEFINE
//===========================================================================
// For GVW margin determination
#define DEF_FOVER           (8000.0)            // GVW determination 8t
#define DEF_F_OV_GEAR2      2.0                 // Excess ratio 2.0 (8t or more)
#define DEF_F_OV_GEAR3      1.7                 // Excess ratio 1.7 (8t or more)
#define DEF_F_OV_GEAR4      1.3                 // Excess ratio 1.3 (8t or more)

#define DEF_F_UN_GEAR2      2.4                 // Excess ratio 2.4 (less than 8t)
#define DEF_F_UN_GEAR3      1.7                 // Excess ratio 1.7 (less than 8t)
#define DEF_F_UN_GEAR4      1.6                 // Excess ratio 1.6(less than 8t)

// GVW normalized engine speed
```

```
#define DEF_F_NE_GEAR2       5                  // Normalized engine speed 5%
#define DEF_F_NE_GEAR3       11                 // Normalized engine speed 11%
#define DEF_F_NE_GEAR4       19                 // Normalized engine speed 19%
#define DEF_F_NE_GEAR5       26                 // Normalized engine speed 26%

#define DEF_MAXGEAR          20                 // Max. gear

#define NB_RIDING_MODE       2
//---------------------------------
// System physical values
//---------------------------------
#define GEAR_HOLD_TIME           3.0     // Gear hold time in internal data (sec)

#define UD                       0.95    // Select optimum gear: Sets final reduction ratio (transmission efficiency) to f
ixed value 0.95.
#define E_FACT                   0.03    // Inertial weight ratio equivalent in rotation section (E_FACT)
#define M_FACT                   0.07    // Inertial weight ratio equivalent in rotation section (M_FACT)
#define OD_BETA_ST               0.60    // Beta strate
#define OD_BETA_SOD              0.81    // Beta single OD
#define OD_BETA_DOD              1.00    // Beta Double OD
#define PERSON_W                 55.0    // Weight per person  (55kg)

#define CLUTCH_RELEASE           4.0     // Sets clutch release normalized engine speed in internal data.
#define CLUTCH_MEET              5.0     // Sets clutch meet normalized revolution in internal data
#define PI                       3.14    // Circle circumference ratio to diameter
#define G                        9.8     // Gravitational acceleration

#define DEF_FORCE_ON98           0.98    // 98%: Gear transmission efficiency
#define DEF_FORCE_OFF95          0.95    // 95%: Gear transmission efficiency

#define DEF_GEAR_RATIO           1
#define DEF_FINAL_REDUC_RATIO    4.711
#define DEF_IDLING_ENGINE_SPEED  500
#define DEF_MAX_OUTPUT_RATIO     3000
#define DEF_LOAD_LIMIT           3100



//---------------------------------
// Values refering to the two modes
//---------------------------------
#define BUS     1
#define TRACTOR 2

#define NB_DIV              25
#define NB_DIV_CATEGORIES   6
extern std::string divNames[NB_DIV];
extern double divValues[NB_DIV][NB_DIV_CATEGORIES];

#define MODE1_SIZE  1830
string nameMode1="URBAN";
extern double mode1[MODE1_SIZE][3];

#define MODE2_SIZE  3120
string nameMode2="HIGHWAY";
extern double mode2[MODE2_SIZE][3];

//---------------------------------
// For console display
//---------------------------------
#define ALLOWED_CAR          "abcdefghijklmnopqrstuvwxyz_ABCDEFGHIJKLMNOPQRSTUVWXYZ.1234567890"

#define X_MAX                           76
#define Y_MAX                           23

//=============================================================================
// Structure/Class
//=============================================================================
typedef struct stData{
    double fTimes;              // Accumulated time  (sec) [for read]
    int    nGear;               // Gear
    int    nGearTime;           // Gear determination time duration (msec)
    bool   bFuelCut;            // Fuel Cut
    double fV;                  // Speed (for pattern data read)
    double fGrade;              // Grade
    double fA;                  // Acceleration (km/sec)
    double fCarA;               // Acceleration (m/msec)
    double fVref_sp;            // Reference speed
    double fVana_sp;            // Analysis speed
    double fNegrevo;            // Engine speed
    double fTe;                 // Engine torque
    double fMaxTe;              // Engine torque Max
    double fFuel;               // Fuel
    double fF;                  // Driving force
}stData;

//-----------------------------
// Excess force ratio data
```

```
//-----------------------------
typedef struct stExF{
    int    nGear;               // Gear position
    double fGearRatio;          // Gear ratio
    double fForcePer;           // Transmission efficiency
    double fFreePer;            // Excess ratio
    double fMinPer;             // Lower-limit engine speed (%-normalized)
    double fMinNe;              // Lower-limit engine speed (calculated)
}stExF;

//-------------------------------------------------------------
// Max. torque data
//-------------------------------------------------------------
typedef struct stMaxTq{
    double fEgtq;               // Engine torque
    double fEgrevo;             // Engine speed
}stMaxTq;

//-----------------------------------------------------------
// Fuel consumption data
//-----------------------------------------------------------
typedef struct stFuelConsum{
    double fEgrevo;             // Engine speed
    double fEgtq;               // Engine torque
    double fEgfuel;             // Engine Fuel
}stFuelConsum;
//-----------------------------------------------------------
// Structure for 1-line data save
//-----------------------------------------------------------
typedef struct stCsvLData{
    vector<string> word;
    vector<int>    type;
}stCsvLData;

//=============================================================================
//  Global Area
//=============================================================================
//---------------------
// input values
//---------------------
string m_SpecFileName;
string m_VehicleName;           // Name given to the vehicle
string m_ConsDivision;          //
int m_iConsDivisionIndex;

string m_EngineFileName;
string m_TransmissionFileName;
double m_fLastReGear;           // Final reduction ratio
double m_fTarR;                 // Tire rolling radius [m]
int    m_nTorqueConv;

bool   m_bOutputResults;

// Caracteristics values
double m_fCarWeight;            // The mass of the car
double m_fGrossVehicleWeight;   //The most weight that a particular vehicle can safely and reliably haul

double m_fCarCurbW;             // Empty vehicle mass (kg)
double m_fCarMaxPayload;        // Max. payload (Kg)

double m_fPersons;              // Riding capacity

double m_fOverHeight;           // Overall vehicle height
double m_fOverWidth;            // Overall vehicle width

int    m_nMaxGear;              // Max. number of gears
int    m_nMaxMainGear;          // Max. number of gears
int    m_nMaxSubGear;           // Max. number of sub-gears

// Values initialized in InitParameters()
double m_fMuRollRes;
double m_fMuAir;
int    m_nCarKind;              // The kind of car (0 : track   1 : bus 2 : tractor)
double m_fCl_RelNe;             // Clutch release engine speed
double m_fCl_MetNe;             // Clutch meet engine speed
double m_fRaOutRot;             // Rated output engine speed [rpm]
double m_fOutRot;               // Loaded limit engine speed [rpm]
double m_fIdleNe;               // Idling (IDLE) engine speed
int    m_nInitGear;             // Starting gear initial value
int    m_nDownLimitGear;        // Down limit gear
double m_fFixedNe;              // Max. engine speed, rated engine speed
bool   m_bLossTqOut;            // Loss-Torque output mode
double m_AveSpeed[3];           // Average Speed
double m_AveFuel[3];            // Average Fuel

string m_OutputData;
double m_fFuelAverageCons;
vector<double> m_vTransmissionData;          // Vector containing specification datas
```

```
map<double,stData>::iterator    p_mpData;    // Analysis data pointer
map< int, map<double,stData> >  mp_PtnData;  // PatternFile
int m_ModeID;
vector<double> m_fGearRatio;                 // Gear ratio
vector<double> m_fMainGearRatio;             // Gear ratio
vector<double> m_fSubGearRatio;              // Sub-Gear ratio
set<stMaxTq>   m_MaxTq;                      // Max. torque data
set<stMaxTq>::iterator p_MaxTq;              // Max. torque data pointer
set<stMaxTq>   m_MaxLossTq;                  // Max. loss-torque data
set<stMaxTq>::iterator p_MaxLossTq;          // Max. loss-torque data pointer
set<stMaxTq>   m_CompMaxTq;                  // Max. torque data (8rpm Complement)
set<stMaxTq>::iterator p_CompMaxTq;          // Max. torque data pointer (8rpm Complement)
set<stMaxTq>   m_CompMaxLossTq;              // Max. loss-torque data (8rpm Complement)
set<stMaxTq>::iterator p_CompMaxLossTq;      // Max. loss-torque data pointer (8rpm Complement)
set<stExF>     m_ExF;                        // Excess force ratio table
set<stExF>::iterator   p_ExF;                // Excess force ratio pointer
set<stExF>::iterator   p_ExFCheck;           // Excess force ratio pointer
set<stExF>::iterator   p_ExFCheckBef;        // Excess force ratio pointer

string m_MaxLossFile;
string m_FuelConsumFile;
vector<string> m_vModeNames;
bool m_bSilentModeON;

//-----------------------------
// Max. torque data operator
//-----------------------------
bool operator<(const stMaxTq& a, const stMaxTq& b ){ return( a.fEgrevo < b.fEgrevo); };

//-----------------------------
// Data operator for excess force ratio
//-----------------------------
bool operator<(const stExF& a, const stExF& b ){return( a.nGear < b.nGear); };

//-----------------------------
// Fuel Consume data operator
//-----------------------------
bool operator<(const stFuelConsum& a, const stFuelConsum& b){ return (a.fEgrevo < b.fEgrevo); };

//-----------------------------
// #includes for CFileIO
//-----------------------------
// Output data (header portion)
#define DEF_PR_POS1          "time(s)"
#define DEF_PR_POS2          "Vtarget(km/h)"
#define DEF_PR_POS3          "Vreal(km/h)"
#define DEF_PR_POS4          "Ne(rpm)"
#define DEF_PR_POS5          "Te(N-m)"
#define DEF_PR_POS6          "N_norm(%)"
#define DEF_PR_POS7          "T_norm(%)"
#define DEF_PR_POS8          "Shift"
#define DEF_PR_POS9          "FC(l/h)"

#define MSGWRITE_ERROR       "File write error." // File write error
#define LINE_MAX_LENGTH      1024    // The number max of caracter per line

// Configuration values
#define ITEM_LENGTH     6
#define LINE_LENGTH     68
#define DIVID_X_MIN     5
#define DIVID_Y_MIN     4

// Deduced values
#define ITEM_NB_PER_LINE    (LINE_LENGTH / ITEM_LENGTH)

#define DIVID_X_MAX   (DIVID_X_MIN + (ITEM_LENGTH*ITEM_NB_PER_LINE) -1)
#define DIVID_Y_MAX   (DIVID_Y_MIN + (NB_DIV/ITEM_NB_PER_LINE))
#define ERR(bSuccess, api) {if (!(bSuccess)) cout<<"Error in "<<api<<" at line "<< __LINE__<<endl;}
#define BACKGROUND_WHITE  (WORD) 0x00f0

/*-------------------------------------------------------------------------*/
//  1.Class Name
//      The reading class of the environment file
//
//  2.Outline
//      An environment file is read and kept.
//
//  3.Function explanation
//      (1)  An environment file is specified.
//      (2)  A pattern file is read.
//      (3)  Various cause files are read.
//      (4)  A torque file is read.
//
//  4.Note
//      None
//
/*-------------------------------------------------------------------------*/
class CFileIO
```

```
{
public:
    CFileIO();                                  // Constructor
    virtual ~CFileIO();                         // Destructor

    bool SaveTransmissionData(string fileName);    // Spec File data read module
    bool SaveTorqueData(string fileName, set<stMaxTq> *p_TorquePointer); // Torque File data read module


    bool WriteCalculateData();

    bool WriteSpecs(FILE *fp);
    bool WriteHead(FILE *fp, int i);
    bool WriteValues(FILE *fp, int i);

    bool FileExists(string filename);           // check File exist module
};


/*---------------------------------------------------------------------------*/
//  1.Class Name
//      Operation class
//
//  2.Outline
//      The class that it faces in the time when it was specified and
//      which does each operation
//
//  3.Function explanation
//      None
//
//  4.Note
//      None
//
/*---------------------------------------------------------------------------*/
class CCalculatePos
{
public:
    CCalculatePos();                                                // Constructor
    virtual ~CCalculatePos();                                       // Destructor

    bool   GetExF(void);                                            // excess force raito setting
    bool   CalcAllData(map<double, stData>::iterator p1);           // Calculate All Data
    bool   CalcTqGear(map<double, stData>::iterator p1, int nGear); // Calculate Torque

    bool   CalcTqV(map<double, stData>::iterator p1, int nGear);

    bool   CalcMaxSp(map<double, stData>::iterator p1, int nGear,
                                                int iMode=0);   // Calculate Max Speed
    double GetLRevMaxTq(double fNe);                                // Calculate Torque (Revise)
    bool   GetLRevMaxLossTq(double fNe, double &fTrq);              // Calculate Loss-Torque (Revise)
    bool   CalcNeGear(map<double, stData>::iterator p1,
                      int nGear, int iMode=0);                      // Calculate Ne
    bool   CalcVanaGear(map<double, stData>::iterator p1,int nGear);// Calculate Vana
    bool   GetGearFullRatio(int nGrear,double &fGearRaito);         // Get Gear Ratio(final reduction ratio)

private:
    bool   GetGearRatio(int nGear,double &fGearRaito);              // Get Gear Ratio
    double GetGearPass(int nGear);                                  // Get Gear transmission efficiency
    double CalcRL(double fV);                                       // Calculate Rolling resistance

    double CalcRS( map<double, stData>::iterator p1 );

    double GetCarRotWeight(int nGear);                              // Rotation partial equivalent mass
};

/*---------------------------------------------------------------------------*/
//  1.Class Name
//      Point reading management gear decision class
//
//  2.Outline
//      By the present state of gear and the future speed,
//      so that it can hold a gear to the point for several seconds the class
//      which a gear is made to be decided as
//
//
//  3.Function explanation
//      (1)  A present position of a calculation is specified.
//      (2)  A gear until the gear holding time is decided.
//
//  4.Note
//      None
//
/*---------------------------------------------------------------------------*/
class CForwardCalc
{
public:
    CForwardCalc();                                     // Constructor
    virtual ~CForwardCalc();                            // Destructor
```

```
    bool CalcGearCheck(map<double, stData>::iterator &p1);   // Calculate gear setting main module

private:
    bool CheckNowShiftChange( map<double, stData>::iterator p1,
                              int nGear, int nOrgGear, int iMode);
    int CheckNTimesHold( map<double, stData>::iterator p1,
                             map<double, stData>::iterator p2,
                             int nGear, int nOrgGear, double &fDiffV, int iMode);
    bool Check3TimesHold( map<double, stData>::iterator p1,
                              int nGear, int nOrgGear, int iMode);
    bool CheckNTimesSpeedHold( map<double, stData>::iterator p1,
                              map<double, stData>::iterator p2,
                              int nGear, int nOrgGear, int &nSetGear, int iMode);
    bool CalcGear( map<double, stData>::iterator p1,
                              int &nGear, int nOrgGear, int iMode);

    bool CalcStFollow(map<double, stData>::iterator &p1);    // Calculate startup gear setting module
    CCalculatePos *pCalPos;
};
/*---------------------------------------------------------------------------*/
//  1.Class Name
//      The reading class of the environment file
//
//  2.Outline
//      An environment file is read and kept.
//
//  3.Function explanation
//      (1)  An environment file is specified.
//      (2)  A pattern file is read.
//      (3)  Various cause files are read.
//      (4)  A torque file is read.
//
//  4.Note
//      None
//
/*---------------------------------------------------------------------------*/
class CConsoleIO
{
public:
    CConsoleIO();                       // Constructor
    virtual ~CConsoleIO();              // Destructor

    void cls();

    void SetCursorPos(int x, int y);
    bool GetCursorPos(int &x, int &y);

    void DisplayMenu();
    void DisplayCaracteristics();
    void DisplayResultsScreen();

    void WriteConsoleError(char * err);
    void myWriteConsole(int curPos_X, int curPos_Y, char * writeValue);


    bool myWriteConsole(HANDLE hConsole, PCHAR s, int x=NULL, int y=NULL);

    int ReadConsole(int curPos_X, int curPos_Y,
                        int maxLength, string &readValue,
                        string allowedCar,
                        bool bFileTest=false);

    // Functions used for Consumption Division election screen
    void ReadConsDivChoice(string &readValue, int &readDivID);
    bool CurPostoDivID(int curPosX, int curPosY, int &consDivID);
    bool CurPostoDivID(COORD curPos, int &consDivID){return CurPostoDivID(curPos.X, curPos.Y, consDivID);};

    void UpdateReadDivScreen(HANDLE hConDivChoice, WORD &PrevAttributes, WORD &ActivAttributes, COORD &curPos, int &consDi
vID);

private:
    bool m_bErrorMessagePrinted;
};

/*---------------------------------------------------------------------------*/
//  1.Class Name
//      A main class
//
//  2.Outline
//      The car speed which data were read in from the environment file
//      given to it and which was analyzed is decided.
//
//  3.Function explanation
//      (1) Various cause files are read.
//      (2) Reading data are computed in the cause.
//      (3) It is output in the file.
//
```

```
//  4.Note
//      None
//
/*-------------------------------------------------------------------------*/
class CExecuteProc
{
public:
    CExecuteProc();                                     // Constructor
    virtual ~CExecuteProc();                            // Destructor

    bool Init(int argc, char* argv[]);                                      // Initialize
    bool CalculateStart(int Index);                     // Calculate start

    bool Init_InputArguments(int argc, char* argv[]);
    bool Init_InputData();                      // Environmental data read module
    bool Init_Caracteristics();


    CFileIO *pReadF;                        // Read file class
    CCalculatePos *pCalPos;                 // Calculate module class
    CForwardCalc *pFCal;                    // Calculate sub class

    CFileIO    *pFileIO;
    CConsoleIO *pConsole;
};


/**/
/*****************************************************************
 * Function name           : CConsoleIO
 * Function summary        : Constructor
 * Explanation             : Class constructor
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
CConsoleIO::CConsoleIO()
{
    return;
}
/**/
/*****************************************************************
 * Function name           : CConsoleIO
 * Function summary        : Constructor
 * Explanation             : Class Destructor
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
CConsoleIO::~CConsoleIO()
{
    return;
}
/**/
/*****************************************************************
 * Function name           : CConsoleIO
 * Function summary        : SetCursorPos
 * Explanation             : SetCursorPos
 *                         :
 * Argument (input)        : int x, y
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
void CConsoleIO::SetCursorPos(int x, int y)
{
#ifdef __GNUC__
    cout<<" \x1b[" << x << ";" << y << "H";
#else
    COORD curPos = {x ,y} ;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), curPos);
#endif
}
/**/
/*****************************************************************
```

```
 * Function name           : CConsoleIO
 * Function summary        : cls
 * Explanation             : cls
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 ****************************************************************/
void CConsoleIO::cls()
{
#ifdef __GNUC__
    cout<<"\x1b[2J";
#else
    COORD coordScreen = { 0, 0 }; /* here's where we'll home the cursor */
    DWORD cCharsWritten;
    CONSOLE_SCREEN_BUFFER_INFO csbi; /* to get buffer info */
    DWORD dwConSize; /* number of character cells in the current buffer */

    /* get the output console handle */
    HANDLE hConsole=GetStdHandle(STD_OUTPUT_HANDLE);
    /* get the number of character cells in the current buffer */
    GetConsoleScreenBufferInfo(hConsole, &csbi);
    dwConSize = csbi.dwSize.X * csbi.dwSize.Y;
    /* fill the entire screen with blanks */
    FillConsoleOutputCharacter(hConsole, (TCHAR) ' ', dwConSize, coordScreen, &cCharsWritten);
    /* get the current text attribute */
    GetConsoleScreenBufferInfo(hConsole, &csbi);
    /* now set the buffer's attributes accordingly */
    FillConsoleOutputAttribute(hConsole, csbi.wAttributes, dwConSize, coordScreen, &cCharsWritten);
    /* put the cursor at (0, 0) */
    SetConsoleCursorPosition(hConsole, coordScreen);
    return;
#endif
}

/**/
/****************************************************************
 * Function name           : CConsoleIO
 * Function summary        : GetCursorPos
 * Explanation             : GetCursorPos
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : int x, y
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 ****************************************************************/
bool CConsoleIO::GetCursorPos(int &x, int &y)
{
    bool bRet=false;

    HANDLE hdl=GetStdHandle(STD_OUTPUT_HANDLE);

    CONSOLE_SCREEN_BUFFER_INFO lpConsoleScreenBufferInfo;
    bRet=GetConsoleScreenBufferInfo(hdl, &lpConsoleScreenBufferInfo);

    if(bRet){
        x=lpConsoleScreenBufferInfo.dwCursorPosition.X;
        y=lpConsoleScreenBufferInfo.dwCursorPosition.Y;
    }

    return bRet;
}
/**/
/****************************************************************
 * Function name           : CConsoleIO
 * Function summary        : DisplayMenu
 * Explanation             : DisplayMenu
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 ****************************************************************/
void CConsoleIO::DisplayMenu()
{
    // Screen is cleared
    cls();
```

```
    cout<<"┌──────────────────────────────────────────────────────────────────────┐"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"│      --= Welcome to Diesel fuel consumption program version " << VERSIONNO <<" ! =--    |"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"│  Please enter the following informations:                            │"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"│   -Vehicle name           :                                          │"<<endl;
    cout<<"│   -Fuel consumption div.  :                                          │"<<endl;
    cout<<"│   -Final Gear ratio       :                                          │"<<endl;
    cout<<"│   -Tire rolling radius(m) :                                          │"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"│   -Engine file name       :                                          │"<<endl;
    cout<<"│   -Transmission file name :                                          │"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"│   -Output results to file? (Y/N):                                    │"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"└──────────────────────────────────────────────────────────────────────┘"<<endl;
}
/**/
/*************************************************************
 * Function name           : CConsoleIO
 * Function summary        : DisplayCaracteristics
 * Explanation             : DisplayCaracteristics
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *************************************************************/
void CConsoleIO::DisplayCaracteristics()
{
    int x=0;
    int y=0;

    int last_x,last_y;   //values used to locate the cursor at the last line

    char   buf[1024];

    // ---------------------------
    // Screen is cleared and cursor position is initialized
    // ---------------------------
    cls();
    GetCursorPos(x,y);


    // ---------------------------
    // Displaying header
    // ---------------------------
    cout<<"┌──────────────────────────────────────────────────────────────────────┐"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"│   -Vehicle caracteristics:                                           │"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"│  Vehicle name :                                                      │"<<endl;
    cout<<"│  Vehicle type :                                                      │"<<endl;
    cout<<"│  Category     :                                                      │"<<endl;
    cout<<"│  Transmission :                                                      │"<<endl;
    cout<<"│  Highway ratio:                                                      │"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"│  GVW  =                                                              │"<<endl;
    cout<<"│  Riding capacity =                                                   │"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"│  Wcurb =             , Wtest =                                       │"<<endl;
    cout<<"│  Width =             , Height=             , Tire radius =           │"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"│  Nidle =             , Nrate =             , Nex =                   │"<<endl;
    cout<<"│  Nes   =             , Nec   =                                       │"<<endl;
    cout<<"│  MuAir =                                                             │"<<endl;
    cout<<"│  MuRoll=                                                             │"<<endl;
    cout<<"│                                                                      │"<<endl;
    cout<<"│  << Next Page >>                                                     │"<<endl;
    cout<<"└──────────────────────────────────────────────────────────────────────┘"<<endl;


    // ---------------------------
    // Vehicle name, type, division, transmission type and highway ridding ratio
    // ---------------------------
    sprintf( buf, "%s", m_VehicleName.c_str());
    myWriteConsole(18, 4, buf);

    if(m_nCarKind==1) sprintf( buf, "Bus");
```

```
else sprintf( buf, "Tractor");
myWriteConsole(18, 5, buf);

sprintf( buf, "%s", m_ConsDivision.c_str());
myWriteConsole(18, 6, buf);

if(m_nTorqueConv==1) sprintf( buf, "Torque Converter AT");
else sprintf( buf, "MT, AMT");
myWriteConsole(18, 7, buf);

sprintf( buf, "%3.2f[%]", divValues[m_iConsDivisionIndex][5]);
myWriteConsole(18, 8, buf);


// --------------------------
// Mass & Ridding capacity
// --------------------------
sprintf( buf, "%8.2f[kg]", m_fGrossVehicleWeight);
myWriteConsole(11, 10, buf);

sprintf( buf, "%3.0f [persons]", m_fPersons);
myWriteConsole(21, 11, buf);


y=13;
// --------------------------
// WCurb & WTest
// --------------------------
sprintf( buf, "%5.2f[kg]", m_fCarCurbW);
myWriteConsole(12, y, buf);

sprintf( buf, "%5.2f[kg]", m_fCarWeight);
myWriteConsole(34, y, buf);

// --------------------------
// Width, Height & Tire radius
// --------------------------
y++;
sprintf( buf, "%3.3f[m]", m_fOverWidth);
myWriteConsole(12, y, buf);

sprintf( buf, "%3.3f[m]", m_fOverHeight);
myWriteConsole(34, y, buf);

sprintf( buf, "%3.3f[m]", m_fTarR);
myWriteConsole(63, y, buf);

// --------------------------
// Nidle, Nrate & Nex
// --------------------------
y+=2;
sprintf( buf, "%5.2f[rpm]", m_fIdleNe);
myWriteConsole(12, y, buf);

sprintf( buf, "%5.2f[rpm]", m_fRaOutRot);
myWriteConsole(34, y, buf);

sprintf( buf, "%5.2f[rpm]", m_fOutRot);
myWriteConsole(55, y, buf);

// --------------------------
// Nes & Nec
// --------------------------
y++;
sprintf( buf, "%5.2f[rpm]", m_fCl_MetNe);
myWriteConsole(12, y, buf);

sprintf( buf, "%5.2f[rpm]", m_fCl_RelNe);
myWriteConsole(34, y, buf);

// --------------------------
// MuAir & MuRoll
// --------------------------
y++;
sprintf( buf, "%10.6f [N/(km/h)^2]", m_fMuAir*G);
myWriteConsole(12, y, buf);

y++;
sprintf( buf, "%10.6f [N/kg]", m_fMuRollRes*G);
myWriteConsole(12, y, buf);

y+=2;
SetCursorPos(20, y);

// --------------------------
// Break During displaying
// --------------------------
if(!m_bSilentModeON)
```

```
    getch();

set<stExF>::iterator p_ExF;         // Excess force ratio data
double fDW;
double fBeta,fGearRatioCheck,fGearRatioCheckbef;
stExF findRatioCheck;

// ---------------------------
// Screen is cleared and cursor position is initialized
// ---------------------------
cls();
GetCursorPos(x,y);

// ---------------------------
// Displaying header
// ---------------------------
cout<<" ┌";                              // top line
for(int i=x; i<X_MAX; i++){
    cout<<"—";
    GetCursorPos(i,y);
}
cout<<"┐ "<<endl; y++; //endl increments the number of lines


// ---------------------------
// Number of gear
// ---------------------------
sprintf( buf, " |  NUMBER OF GEAR = %2d", m_nMaxGear );
cout << buf; SetCursorPos(X_MAX,y); cout<<" | "<<endl;  y++;


// ---------------------------
// Gear - ratio - efficiency - torque - margin - DW
// ---------------------------
sprintf( buf, " |  GEAR   RATIO  EFFICIENCY  TORQ MARGIN     DW[kg]");
cout << buf; SetCursorPos(X_MAX,y); cout<<" | "<<endl;  y++;

findRatioCheck.nGear = m_nMaxGear;
p_ExFCheck = m_ExF.find(findRatioCheck);
findRatioCheck.nGear = m_nMaxGear-1;
p_ExFCheckBef = m_ExF.find(findRatioCheck);

for(p_ExF = m_ExF.begin();p_ExF != m_ExF.end();p_ExF++ ){
    if(( m_ConsDivision == "TT1" )||( m_ConsDivision == "T10" )||( m_ConsDivision == "T11" )){
        if(( m_nMaxSubGear == 1 )&&( m_nMaxMainGear == 7 )){
            if( p_ExFCheck->fGearRatio == 1.0 ){
                fBeta = OD_BETA_ST;
            }else if( p_ExFCheckBef->fGearRatio == 1.0 ){
                fBeta = OD_BETA_SOD;
            }else{
                fBeta = OD_BETA_DOD;
            }
        }else if(( m_nMaxSubGear == 2 )&&( m_nMaxMainGear == 6 )){
            if( p_ExFCheck->fGearRatio == 1.0 ){
                fBeta = OD_BETA_ST;
            }else{
                fBeta = OD_BETA_DOD;
            }
        }else{
            fBeta = OD_BETA_DOD;
        }
    }else{
        fBeta = OD_BETA_DOD;
    }

    fDW = (M_FACT + E_FACT * (p_ExF->fGearRatio * fBeta) * (p_ExF->fGearRatio * fBeta)) * m_fCarCurbW;

    sprintf( buf, " |  %3d:   %6.3f    %6.3f    %6.3f   %12.5f",
                  p_ExF->nGear,
                  p_ExF->fGearRatio,
          p_ExF->fForcePer,
          p_ExF->fFreePer,
          fDW );
    cout << buf; SetCursorPos(X_MAX,y); cout<<" | "<<endl;  y++;
}
sprintf( buf, " |  FIN:   %6.3f    %6.3f", m_fLastReGear, UD );
cout << buf; SetCursorPos(X_MAX,y); cout<<" | "<<endl;  y++;


// ---------------------------
// Displaying footer
// ---------------------------
while(y<Y_MAX-2){
    cout<<" | "; SetCursorPos(X_MAX,y); cout<<" | "<<endl;  y++;// blank line
}
cout<<" |  << Next Page >>";GetCursorPos(last_x,last_y);SetCursorPos(X_MAX,y); cout<<" | "<<endl;  y++;
cout<<" | "; SetCursorPos(X_MAX,y); cout<<" | "<<endl;  y++;// blank line
```

```
    cout<<" └";                              // top line
    for(int i=x; i<X_MAX; i++){
        cout<<"──";
        GetCursorPos(i,y);
    }
    cout<<"┘ ";
    SetCursorPos(last_x,last_y);
}
/**/
/*****************************************************************
 * Function name           : CConsoleIO
 * Function summary        : DisplayResultsScreen
 * Explanation             :
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
void CConsoleIO::DisplayResultsScreen()
{
    // Screen is cleared
    cls();

    cout<<" ┌──────────────────────────┬──────────────────────────┬──────────────────────────┐                                                  "<<endl;
    cout<<" │                          │                          │                          │                                                  "<<endl;
    cout<<" │  MODE1=                  │  MODE2=                  │  MODE3= MID-TOWN         │                                                  "<<endl;
    cout<<" │                          │                          │                          │                                                  "<<endl;
    cout<<" │  Vave =                  │  Vave =                  │  Vave =                  │                                                  "<<endl;
    cout<<" │  FC =                    │  FC =                    │  FC =                    │                                                  "<<endl;
    cout<<" ├──────────────────────────┴──────────────────────────┴──────────────────────────┘                                                  "<<endl;
    cout<<" │                                                                                                                                  "<<endl;
    cout<<" │  Average Fuel comsumption =                                                                                                      "<<endl;
    cout<<" │                                                                                                                                  "<<endl;
    cout<<" │  Output File :                                                                                                                   "<<endl;
    cout<<" │                                                                                                                                  "<<endl;
    cout<<" │                                                                                                                                  "<<endl;
    cout<<" │                                                                                                                                  "<<endl;
    cout<<" │                                                                                                                                  "<<endl;
    cout<<" │                                                                                                                                  "<<endl;
    cout<<" │                                                                                                                                  "<<endl;
    cout<<" │                                                                                                                                  "<<endl;
    cout<<" └──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────"<<endl;
}
/**/
/*****************************************************************
 * Function name           : CConsoleIO
 * Function summary        : WriteConsoleError
 * Explanation             :
 *                         :
 * Argument (input)        : char *err
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
void CConsoleIO::WriteConsoleError(char * err)
{
    m_bErrorMessagePrinted=true;

    myWriteConsole(0, 17, " |                                                                                                                |");
    myWriteConsole(4, 17, err);

    DWORD cCharsWritten;
    COORD curPos={4, 17};
    FillConsoleOutputAttribute(GetStdHandle(STD_OUTPUT_HANDLE), FOREGROUND_INTENSITY¦FOREGROUND_RED¦BACKGROUND_WHITE, strl
en(err), curPos, &cCharsWritten);
}
/**/
/*****************************************************************
 * Function name           : CConsoleIO
 * Function summary        : myWriteConsole
 * Explanation             :
 *                         :
 * Argument (input)        : int x, int y, char *Value
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
```

```
void CConsoleIO::myWriteConsole(int curPos_X, int curPos_Y, char * writeValue)
{
    SetCursorPos(curPos_X, curPos_Y);
    cout<<writeValue;
}
/**/
/*****************************************************************
 * Function name           : CConsoleIO
 * Function summary        : myWriteConsole
 * Explanation             :
 *                         :
 * Argument (input)        : Handle hcon, PCHAR s, int x, int y
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool CConsoleIO::myWriteConsole(HANDLE hConsole, PCHAR s, int x, int y)
{
    BOOL bSuccess;
    DWORD cCharsWritten;

    if(x!=0 && y!=0){
        COORD tmp = {x,y};
        SetConsoleCursorPosition(hConsole, tmp);
    }


    // Write the string to the console
    bSuccess = WriteConsole(hConsole, s, strlen(s), &cCharsWritten, NULL);
    ERR(bSuccess, "myWriteConsole(HANDLE, PCHAR, COORD)");

  return(bSuccess);
}

void DivIDtoCurPos(int consDivID, COORD &curPos)    // consDivID==> 0->(NB_DIV-1)
{
    // Checking validity of consDivID
    if(consDivID<0 || consDivID>=NB_DIV)
        return;

    curPos.X = DIVID_X_MIN + (consDivID%ITEM_NB_PER_LINE) * ITEM_LENGTH;
    curPos.Y = DIVID_Y_MIN + (consDivID/ITEM_NB_PER_LINE);
}

/**/
/*****************************************************************
 * Function name           : CConsoleIO
 * Function summary        : CurPostoDivID
 * Explanation             :
 *                         :
 * Argument (input)        : int x, int y
 * Argument (output)       : None
 * Argument (I/O)          : int consDivID
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool CConsoleIO::CurPostoDivID(int curPosX, int curPosY, int &consDivID)// consDivID==> 0->(NB_DIV-1)
{
    int tmpDivID;

    // Checking validity of curPos
    if(curPosX < DIVID_X_MIN || curPosX > DIVID_X_MAX || curPosY < DIVID_Y_MIN || curPosY > DIVID_Y_MAX)
       return false;

    tmpDivID = (curPosY-DIVID_Y_MIN)*ITEM_NB_PER_LINE;
    tmpDivID+= (curPosX-DIVID_X_MIN)/ITEM_LENGTH;

    // Division ID must be between 0 and (NB_DIV-1)
    if(tmpDivID>=0 && tmpDivID<NB_DIV)
        consDivID = tmpDivID;

    return true;
}
/**/
/*****************************************************************
 * Function name           : CConsoleIO
 * Function summary        : UpdateReadDivScreen
 * Explanation             :
 *                         :
 * Argument (input)        : Handle
 * Argument (output)       : None
 * Argument (I/O)          : WORD Attrib, WORD Attrib COORD &curPos, int &consDivID
 * Return value            : None
```

```
 * Created by             :
 * Updated on (created on) :
 * Remarks                 :
 ****************************************************************/
void CConsoleIO::UpdateReadDivScreen(HANDLE hConDivChoice, WORD &PrevAttributes, WORD &ActivAttributes, COORD &curPos, int
 &consDivID)
{
    DWORD cCharsWritten;

    //Restore previous highlighted item
    FillConsoleOutputAttribute(hConDivChoice, PrevAttributes, ITEM_LENGTH, curPos, &cCharsWritten);

    //Highlight active index
    DivIDtoCurPos(consDivID,curPos);
    FillConsoleOutputAttribute(hConDivChoice, ActivAttributes, ITEM_LENGTH, curPos, &cCharsWritten);


    CHAR buff[256];

    sprintf( buff, "%5.1f [kg]                ", 1000*divValues[consDivID][0]);
    myWriteConsole(hConDivChoice, buff, 17, 11);

    sprintf( buff, "%5.1f [kg]                ", 1000*divValues[consDivID][1]);
    myWriteConsole(hConDivChoice, buff, 17, 12);

    sprintf( buff, "%.0f [persons]                ",  divValues[consDivID][2]);
    myWriteConsole(hConDivChoice, buff, 17, 13);

    sprintf( buff, "%3.3f [m]                ",       divValues[consDivID][3]);
    myWriteConsole(hConDivChoice, buff, 17, 14);

    sprintf( buff, "%3.3f [m]                ",       divValues[consDivID][4]);
    myWriteConsole(hConDivChoice, buff, 17, 15);

    sprintf( buff, "%3.2f [%]                ",       divValues[consDivID][5]);
    myWriteConsole(hConDivChoice, buff, 17, 16);

    COORD tmpCoord={50,19};
    SetConsoleCursorPosition(hConDivChoice, tmpCoord);
}
/**/
/****************************************************************
 * Function name           : CConsoleIO
 * Function summary        : ReadConsDivChoice
 * Explanation             :
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : string Value, int readDivID
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 ****************************************************************/
void CConsoleIO::ReadConsDivChoice(string &readValue, int &readDivID)
{
    HANDLE hConOld=GetStdHandle(STD_OUTPUT_HANDLE);

    BOOL bSuccess;
    HANDLE hConDivChoice; /* console for the new screen */

    DWORD cCharsWritten;
    CHAR c = 0; /* virtual key code that we will read */
    HANDLE hConCurrent; /* keep track of the current visible console buffer */
    INPUT_RECORD inputBuf; /* console input event record */
    DWORD cInputEvents;

    /* create a separate console buffer for the new screen */
    hConDivChoice = CreateConsoleScreenBuffer(GENERIC_READ | GENERIC_WRITE,
      FILE_SHARE_READ | FILE_SHARE_WRITE, NULL, CONSOLE_TEXTMODE_BUFFER,
      NULL);
    ERR(hConDivChoice != INVALID_HANDLE_VALUE, "ReadConsDivChoice");

    myWriteConsole(hConDivChoice,
    "                                                                  \n"
    "  Select here a consumption division:                             \n"
    "                                                                  \n"
    "                                                                  \n"
    "                                                                  \n"
    "                                                                  \n"
    "                                                                  \n"
    "                                                                  \n"
    "                                                                  \n"
    "                                                                  \n"
    "  WCurb       =                                                   \n"
    "  MaxPayload  =                                                   \n"
    "  Crew        =
```

```
    "  Height      =                                                                  \n"
    "  Width       =                                                                  \n"
    "  HighWayRatio=                                                                  \n"
    "                                                                                 \n"
    "                                                                                 \n"
    "  << Press enter or click to select a division >>                                \n"
    "                                                                                 \n"
    "                                                                                 \n");


    COORD curPos={DIVID_X_MIN, DIVID_Y_MIN};
    char tmp[ITEM_LENGTH+1], buf[ITEM_LENGTH];
    int  consDivID;

    for(consDivID=0; consDivID<NB_DIV ; consDivID++)
    {
        sprintf( tmp, " %s    ", divNames[consDivID].c_str());
        strncpy(buf, tmp, ITEM_LENGTH);
        buf[sizeof(buf)-1]='\0';

        DivIDtoCurPos(consDivID, curPos);

        SetConsoleCursorPosition(hConDivChoice, curPos);
        myWriteConsole(hConDivChoice, buf);
    }


    // Setting actual and new display mode
    CONSOLE_SCREEN_BUFFER_INFO csbi;
    GetConsoleScreenBufferInfo(hConDivChoice, &csbi);

    WORD PrevAttributes = csbi.wAttributes;
    WORD ActivAttributes= FOREGROUND_INTENSITY|FOREGROUND_RED|BACKGROUND_WHITE;

    //Default active consDivID is 0
    consDivID=0;

    // Prepare display of first values
    DivIDtoCurPos(consDivID, curPos);
    UpdateReadDivScreen(hConDivChoice, PrevAttributes, ActivAttributes, curPos, consDivID);


    //-------------------------
    // Activation of the new console:
    //-------------------------
    SetConsoleActiveScreenBuffer(hConDivChoice);

    //#########################################################################
    // Reading keyboard and mouse events
    //#########################################################################
    HANDLE hStdIn; /* standard input handle */
    hStdIn = GetStdHandle(STD_INPUT_HANDLE);
    ERR(hStdIn != INVALID_HANDLE_VALUE, "GetStdHandle");

    for(;;)
    {
    do{
        /* throw away any non-keystroke events or any key-up events */
        bSuccess = ReadConsoleInput(hStdIn, &inputBuf, 1, &cInputEvents);
        ERR(bSuccess, "ReadConsoleInput");
    }
    while (!((inputBuf.EventType == MOUSE_EVENT) || (inputBuf.EventType == KEY_EVENT && inputBuf.Event.KeyEvent.bKeyDown)))
;

    switch(inputBuf.EventType){
        case MOUSE_EVENT:
            // In case of mouse click
            if(inputBuf.Event.MouseEvent.dwEventFlags == 0){
                if(consDivID>=0 && consDivID<NB_DIV){
                    readValue = divNames[consDivID];
                    readDivID = consDivID;
                    SetConsoleActiveScreenBuffer(hConOld);
                    return;
                }
                break;
            }
            //Upgrading index of active item
            bSuccess = CurPostoDivID(inputBuf.Event.MouseEvent.dwMousePosition.X,
                          inputBuf.Event.MouseEvent.dwMousePosition.Y,
                          consDivID);

            if(bSuccess)
                UpdateReadDivScreen(hConDivChoice, PrevAttributes, ActivAttributes, curPos, consDivID);
            break;

        case KEY_EVENT:
            /* get the virtual scan code of the key-down event */
            c = (char) inputBuf.Event.KeyEvent.wVirtualKeyCode;
```

```
            switch(c){
                case VK_LEFT:
                    if(consDivID>0){
                        consDivID--;

                        UpdateReadDivScreen(hConDivChoice, PrevAttributes, ActivAttributes, curPos, consDivID);
                    }
                    break;

                case VK_RIGHT:
                    if((consDivID+1)<NB_DIV){
                        consDivID++;

                        UpdateReadDivScreen(hConDivChoice, PrevAttributes, ActivAttributes, curPos, consDivID);
                    }
                    break;

                case VK_UP:
                    if( consDivID-ITEM_NB_PER_LINE>=0){
                        consDivID-=ITEM_NB_PER_LINE;

                        UpdateReadDivScreen(hConDivChoice, PrevAttributes, ActivAttributes, curPos, consDivID);
                    }
                    break;

                case VK_DOWN:
                    if( consDivID+ITEM_NB_PER_LINE < NB_DIV){
                        consDivID+=ITEM_NB_PER_LINE;

                        UpdateReadDivScreen(hConDivChoice, PrevAttributes, ActivAttributes, curPos, consDivID);
                    }
                    break;

                case VK_RETURN:
                    if(consDivID>=0 && consDivID<NB_DIV){
                        readValue = divNames[consDivID];
                        readDivID = consDivID;
                        SetConsoleActiveScreenBuffer(hConOld);
                        return;
                    }
                    break;
                default: break;
            }

        default: break;
    }  /* switch*/
    }  /* for */
}
/**/
/*****************************************************************
 * Function name           : CConsoleIO
 * Function summary        : ReadConsole
 * Explanation             :
 *                         :
 * Argument (input)        : int x, int y, int len, string allow, bool bTest
 * Argument (output)       : string Value
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
int CConsoleIO::ReadConsole(int curPos_X, int curPos_Y, int maxLength, string &readValue, string allowedCar, bool bFileTes
t)
{
    int key;
    int x=0;    // Relative position of the cursor
    bool bInsertMode=false;
    int outputVal=0;

    int ValidatedCar=0;

    x=readValue.length();
    SetCursorPos(curPos_X+x, curPos_Y);

    char tmpReadV[128];
    strncpy(tmpReadV, readValue.c_str(), sizeof(tmpReadV));
    myWriteConsole(curPos_X, curPos_Y, tmpReadV);
    SetCursorPos(curPos_X+x, curPos_Y);


LISTEN:
    do{
        key=getch();

        // A valid key as been pushed so error message, if any, is reseted
        if(m_bErrorMessagePrinted){
            int tmpX=0, tmpY=0;
```

```
        GetCursorPos(tmpX, tmpY);
        myWriteConsole(0, 17, "|                                                                                                  |");
        m_bErrorMessagePrinted=false;
        SetCursorPos(tmpX, tmpY);
    }

    if(key==0){
        key=getch();

        switch(key){

        case 72: return -1;                // UP

        case 75: //LEFT ARROW
            if(x>0){
                SetCursorPos(--x + curPos_X, curPos_Y);
                bInsertMode=true;
            }
            break;

        case 77: //RIGHT ARROW
            if(x<maxLength && x<(int)readValue.length()){
                SetCursorPos(++x + curPos_X, curPos_Y);
            }
            bInsertMode=x<maxLength && x<(int)readValue.length();
            break;

        case 83: //DELETE
            if(bInsertMode){
                if(x+1<(int)readValue.length()){
                    strcpy(&readValue.at(x),&readValue.at(x+1));
                    readValue.resize(readValue.length()-1);
                    cout<<readValue.substr(x, readValue.length()-x+1).c_str()<<" ";
                }
                else{
                    readValue.resize(readValue.length()-1);
                    cout<<" ";
                    bInsertMode=false;
                }
                SetCursorPos(x + curPos_X, curPos_Y);
            }
        }
        continue;
    }

    else if(key==8){ // BACKSPACE
        if(x>0){
            if(bInsertMode){
                strcpy(&readValue.at(x-1),&readValue.at(x));
                readValue.resize(readValue.length()-1);
                cout<<"\b"<<readValue.substr(x-1, readValue.length()-x+1).c_str()<<" ";
            }
            else {
                readValue.resize(readValue.length()-1);
                cout<<"\b ";
            }
            SetCursorPos(--x + curPos_X, curPos_Y);
        }
    }

    else
    {
        ValidatedCar=allowedCar.find((char)key);
        if((int)readValue.length()<maxLength && ValidatedCar>=0)
        {
            if(bInsertMode)
            {
                string s = "";
                s+=(char)key;
                readValue.insert(x, s);
                cout<<readValue.substr(x, readValue.length()-x).c_str();
            }
            else
            {
                readValue+=(char)key;
                 cout<<(char)key;
            }
            SetCursorPos(++x + curPos_X, curPos_Y);
        }
    }
}
while( !(key == 13 && !readValue.empty()) );


//--------------------------------
// File test
//--------------------------------
//If a file test is asked:
```

```
    if(bFileTest){
        FILE *fp;
        string cpy_readValue = readValue;

        fp = fopen( readValue.c_str(), "r");
        if(( fp == NULL )||( ferror(fp) )){
            char tmpErrMessage[256];
            if(cpy_readValue.size()>28){
                cpy_readValue.resize(28);
                cpy_readValue+="(...)";
            }
            sprintf(tmpErrMessage, "Error! : The file '%s' could not be read.", cpy_readValue.c_str());
            WriteConsoleError(tmpErrMessage);
            SetCursorPos(x+curPos_X, curPos_Y);

            outputVal=0;
            goto LISTEN;
        }
        fclose(fp);
    }
    return outputVal;
}
/**/
/*****************************************************************
 * Function name           : CExecuteProc
 * Function summary        : Init_InputArguments
 * Explanation             :
 *                         :
 * Argument (input)        : int argc, char *argv[]
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : true : Normal  false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool CExecuteProc::Init_InputArguments(int argc, char* argv[])
{
    bool bDivModeFound=false;
    bool bConsolePrompt=false;

    m_bSilentModeON=false;

    if( argc >=3 ){
        m_SpecFileName = string(argv[1]);
        if(!pFileIO->FileExists(m_SpecFileName))
            bConsolePrompt=true;
        else{

            char tmp_lineRead[LINE_MAX_LENGTH];
            char *p;

            FILE *fp_specfile = fopen(m_SpecFileName.c_str() , "r" );
            if((fp_specfile == NULL)||(ferror(fp_specfile)))
                bConsolePrompt=true;
            //-------------------------------
            //Reading the file and storing elements in structure
            //-------------------------------
            for(int row=1; fgets(tmp_lineRead, LINE_MAX_LENGTH, fp_specfile); ){

                //Rotation Number
                p = strtok(tmp_lineRead, "  \t,;%\n");
                if( p == NULL ) continue;//Incomplete or empty line is ignored

                switch (row){
                    case 1:m_VehicleName   = string(p);break;
                    case 2:m_ConsDivision  = string(p);break;
                    case 3:m_EngineFileName= string(p);break;
                    case 4:m_TransmissionFileName = string(p);break;
                    case 5:m_fLastReGear = atof(p);break;
                    case 6:m_fTarR = atof(p);break;
                    case 7:m_bOutputResults = (atoi(p)==1);break;

                    default:break;
                }
                row++;
            }
            //----------------------
            // Checking the correct value of Division
            //----------------------
            for(int i=0;i<m_ConsDivision.length();i++)
                m_ConsDivision[i] = toupper(m_ConsDivision[i]);
            for(int i=0; (i<NB_DIV && !bDivModeFound); i++){  // presence is checked
                if(m_ConsDivision==divNames[i]){
                    m_iConsDivisionIndex=i;
                    bDivModeFound=true;
                }
            }
```

```
            if(!bDivModeFound) bConsolePrompt=true;

            //----------------------
            // Checking presence of engine file
            //----------------------
            if(!pFileIO->FileExists(m_EngineFileName))
                bConsolePrompt=true;
        }

        // Retreives Output file name
        if(argc >=3) m_OutputData = string(argv[2]);
        else m_OutputData="";

        // Detecting silent mode or not
        if(argc >=4){
            if(strcmp(argv[3],"-s")==0 || strcmp(argv[3],"-S")==0){
                m_bSilentModeON=true;
            }
        }
    }

    //----------------------
    // If arguments are not present or not complete data are read via console prompt
    //----------------------
    if(argc <2 || bConsolePrompt){
        string tmpStr;
        READ_INPUT_DATA_AGAIN:

        //reseting values
        m_VehicleName="";
        m_ConsDivision="";
        string str_fLastReGear="";
        string str_fTarR="";
        m_EngineFileName="";
        m_TransmissionFileName="";
        string str_bOutputResults="";

        int res;

        pConsole->DisplayMenu();

        //----------------------
        // 1) Gets vehicle name
        //----------------------
READ_1:
        res=pConsole->ReadConsole(31, 7,45, m_VehicleName, ALLOWED_CAR);
        if(res==-1) goto READ_1;

        //----------------------
        // 2) Gets consumption division
        //----------------------
READ_2:
        res=pConsole->ReadConsole(31, 8, 3, m_ConsDivision, "?brtBRT1234567890");
        if(res==-1) goto READ_1;
        bDivModeFound=false;
        if(m_ConsDivision[0]=='?')
        {
            pConsole->ReadConsDivChoice(m_ConsDivision, m_iConsDivisionIndex);
            char buf[256];
            sprintf( buf, "%s", m_ConsDivision.c_str());
            pConsole->myWriteConsole(31, 8, buf);
        }

        else{
            for(int i=0;i<m_ConsDivision.length();i++)
                m_ConsDivision[i] = toupper(m_ConsDivision[i]);
            for(int i=0; i<NB_DIV && !bDivModeFound; i++){
                if( m_ConsDivision ==divNames[i]){
                    m_iConsDivisionIndex=i;
                    bDivModeFound=true;
                }
            }
            if(!bDivModeFound){
                char tmpErrMessage[256];
                sprintf(tmpErrMessage, "Error! : The division mode you entered ('%s') was unknown.", m_ConsDivision.c_str
());
                pConsole->WriteConsoleError(tmpErrMessage);
                pConsole->myWriteConsole(31, 8, "   ");
                goto READ_2;
            }
        }

        //----------------------
        // 3) Gets final gear ratio
        //----------------------
READ_3:
        tmpStr=str_fLastReGear;
        res=pConsole->ReadConsole(31, 9, 6, tmpStr, "1234567890.");str_fLastReGear=tmpStr;
```

```
        m_fLastReGear=atof(tmpStr.c_str());
        if(res==-1) goto READ_2;

        //----------------------
        // 4) Gets tire rolling radius
        //----------------------
READ_4:
        tmpStr=str_fTarR;
        res=pConsole->ReadConsole(31,10, 6, tmpStr, "1234567890."); str_fTarR=tmpStr;
        m_fTarR=atof(tmpStr.c_str());
        if(res==-1) goto READ_3;

        //----------------------
        // 5) Gets engine file name
        //----------------------
READ_5:
        res=pConsole->ReadConsole(31,12, 45, m_EngineFileName, ALLOWED_CAR, true);
        if(res==-1) goto READ_4;

        //----------------------
        // 6) Gets transmission file name
        //----------------------
READ_6:
        res=pConsole->ReadConsole(31,13,45, m_TransmissionFileName, ALLOWED_CAR, true);
        if(res==-1) goto READ_5;

        //----------------------
        // 7) Does results have to be written into output file?
        //----------------------
READ_7:
        tmpStr=str_bOutputResults;
        res=pConsole->ReadConsole(35, 15, 1, tmpStr, "ynYN"); str_bOutputResults=tmpStr;
        m_bOutputResults=(tmpStr=="y" || tmpStr=="Y");
        if(res==-1) goto READ_6;

        pConsole->myWriteConsole(2, 18, "->Conversion process will now start..." );
        pConsole->myWriteConsole(3, 19, "Press 'Enter' to start process ('Esc' to re-enter values)" );

        int key=0;
        do{
            key=getch();
        }
        while(key!=27 && key!=13);

        if(key==27) goto READ_INPUT_DATA_AGAIN;
    }

    return true;
}
/**/
/*****************************************************************
 * Function name           : CExecuteProc
 * Function summary        : Init_InputData
 * Explanation             : Init_InputData
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : true : Normal  false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool CExecuteProc::Init_InputData()
{
    bool bRet;
    FILE *fp_file;   // Pointer to the main environment file
    char tmp_lineRead[LINE_MAX_LENGTH];
    memset(tmp_lineRead, 0x00, LINE_MAX_LENGTH ); //Reset of tmp_lineRead
    char *p;

    string s_EngineFileNames[6];

    int nbLines=0;

    // ***********************************************//
    //                READING Engine file             //
    // ***********************************************//
    //--------------------------------
    // A- Opens the m_EngineFileName ; process is stopped if file is not found
    //--------------------------------
    if(!pFileIO->FileExists(m_EngineFileName )){
        cout << "File Not Found. [" << m_EngineFileName.c_str() << "]" << endl;
        return false;
    }
    fp_file = fopen( m_EngineFileName.c_str(), "r" );
    if((fp_file == NULL)||(ferror(fp_file))){
        return false;
```

```
    }

    //------------------------------
    // B- Reads the m_EngineFileName file and stores its content in s_EngineFileNames array
    //------------------------------
    while(!feof(fp_file) && nbLines<6){
        fgets(tmp_lineRead, LINE_MAX_LENGTH, fp_file);
        p= strtok(tmp_lineRead, "  \r \t \n");  // Delimiters are " ", " \r", " \t" and " \n"
        if(p == NULL) continue;

        s_EngineFileNames[nbLines]=string(p);
        nbLines++;
    }
    fclose(fp_file);

    // ----------------------------------------
    // 1) Saving FUEL CONSUMPTION MAPPING file name (And check if it exists)
    //----------------------------------------
    int i=0;
    m_FuelConsumFile = s_EngineFileNames[0];

    // Verifiy if file exists (Content is not read)
    if(!pFileIO->FileExists(m_FuelConsumFile)){
        cout << "File Not Found. [" << m_FuelConsumFile.c_str() << "]" << endl;
        return false;
    }

    // ----------------------------------------
    // 2) Saving TORQUE data
    //----------------------------------------
    i++;
    if(!pFileIO->FileExists(s_EngineFileNames[i])){
        cout << "File Not Found. [" << s_EngineFileNames[i] << "]" << endl;
        return false;
    }
    bRet = pFileIO->SaveTorqueData( s_EngineFileNames[i], &m_MaxTq );
    if(!bRet) return false;

    // ----------------------------------------
    // 3) Saving LOSS TORQUE data
    //----------------------------------------
    i++;
    m_MaxLossFile = s_EngineFileNames[i];
    m_bLossTqOut=true;
    if(!pFileIO->FileExists(s_EngineFileNames[i])){
        m_bLossTqOut = false;
    }
    bRet = pFileIO->SaveTorqueData( s_EngineFileNames[i] , &m_MaxLossTq );
    if(!bRet) return false;

    // ----------------------------------------
    // 4) Idling speed
    //----------------------------------------
    i++;
    m_fIdleNe=atof(s_EngineFileNames[i].c_str());

    // ----------------------------------------
    // 5) Maximum output speed
    //----------------------------------------
    i++;
    m_fRaOutRot=atof(s_EngineFileNames[i].c_str());
    m_fFixedNe = m_fRaOutRot;

    // ----------------------------------------
    // 6) Maximum engine speed
    //----------------------------------------
    i++;
    m_fOutRot=atof(s_EngineFileNames[i].c_str());


    // ************************************************//
    //                Transmission file                //
    // ************************************************//
    bRet = pFileIO->SaveTransmissionData(m_TransmissionFileName);
    if(!bRet) return false;

    return true;
}

/**/
/*************************************************************
 * Function name           : SaveTransmissionData
 * Function summary        : Copies specification datas in locale vector
 * Explanation             :
 *                         :
 * Argument (input)        : string filename
 * Argument (output)       : None
 * Argument (I/O)          : None
```

```
 * Return value           : 1: success ; others:failure, an error code is returned
 * Created by             :
 * Updated on (created on) :
 * Remarks                :
 ****************************************************************/
bool CFileIO::SaveTransmissionData(string fileName)
{
    FILE *fp_file;
    char *p;
    char tmp_lineRead[LINE_MAX_LENGTH];
    memset(tmp_lineRead, 0x00, LINE_MAX_LENGTH ); //Reset of tmp_lineRead
    double d_tmpSpecValue;

    //--------------------------------
    //Opens and Reads the Envfile
    //--------------------------------
    fp_file = fopen(fileName.c_str() , "r" );
    if((fp_file == NULL)||(ferror(fp_file)))
        return false;

    //--------------------------------
    //Reading the file and storing the specifications value
    //--------------------------------
    for(int row=0; fgets(tmp_lineRead, LINE_MAX_LENGTH, fp_file); ){
        //If not number, line is not read
        if(((int)tmp_lineRead[0] < 0x30) || ((int)tmp_lineRead[0])> 0x39)
                continue;

        p = strtok(tmp_lineRead, "  \t \n");
        m_vTransmissionData.push_back(atof(p));

        row++;
    }
    fclose(fp_file);

    //--------------------------------
    // Check validity of datas
    //--------------------------------
    if(m_vTransmissionData.empty())
        return false;

    int nbMainGears=(int)m_vTransmissionData[1];
    int nbSubGears=(int)m_vTransmissionData[2+m_vTransmissionData[1]];

    if((int)m_vTransmissionData.size()!=(1+1+nbMainGears+1+nbSubGears+1))
        return false;

    return true;
}

/**/
/****************************************************************
 * Function name          : CFileIO
 * Function summary       : Constructor
 * Explanation            : Class constructor
 *                        :
 * Argument (input)       : None
 * Argument (output)      : None
 * Argument (I/O)         : None
 * Return value           : None
 * Created by             :
 * Updated on (created on) :
 * Remarks                :
 ****************************************************************/
CFileIO::CFileIO()
{
    return;
}
/**/
/****************************************************************
 * Function name          : ~CFileIO
 * Function summary       : Destructor
 * Explanation            : Class destructor
 *                        :
 * Argument (input)       : None
 * Argument (output)      : None
 * Argument (I/O)         : None
 * Return value           : None
 * Created by             :
 * Updated on (created on) :
 * Remarks                :
 ****************************************************************/
CFileIO::~CFileIO()
{
    return;
}

/**/
```

```
/*****************************************************************
 * Function name           : FileExists
 * Function summary        : File exists check module
 * Explanation             : check file is alone or not
 *                         :
 * Argument (input)        : filename : check file name
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : true : alone  false : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool CFileIO::FileExists( string filename )
{
    FILE *fp;

    fp = fopen( filename.c_str(), "r");
    if(( fp == NULL )||( ferror(fp) )){
        return false;
    }
    fclose(fp);

    return true;
}
/**/
/*****************************************************************
 * Function name           : SaveTorqueData
 * Function summary        : Max. torque data setup processing
 * Explanation             : Data is read from max. torque data file.
 *                         :
 * Argument (input)        : fileName : Max. torque data file name
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : true : Normal  false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool CFileIO::SaveTorqueData( string fileName, set<stMaxTq> * p_TorquePointer )
{
    FILE *fp_torquefile;
    stMaxTq tmpMaxTq;
    char *p;
    char tmp_lineRead[LINE_MAX_LENGTH];
    memset(tmp_lineRead, 0x00, LINE_MAX_LENGTH ); //Reset of tmp_lineRead

    //--------------------------------
    //Opens and Reads the Envfile
    //--------------------------------
    fp_torquefile = fopen(fileName.c_str() , "r" );
    if((fp_torquefile == NULL)||(ferror(fp_torquefile)))
        return false;


    // --------------------------------
    // Deletes existing data if any.
    // --------------------------------
    if (p_TorquePointer->empty() != true){                          // If data exists
        p_TorquePointer->erase( p_TorquePointer->begin(), p_TorquePointer->end() );    // Deletes existing data.
        p_TorquePointer->clear();
    }

    //--------------------------------
    //Reading the file and storing elements in structure
    //--------------------------------

    for(int row=0; fgets(tmp_lineRead, LINE_MAX_LENGTH, fp_torquefile); row++){
        if(row<2) continue;

        //Rotation Number
        p = strtok(tmp_lineRead, "  \t,;% \n");
        if( p == NULL ) continue;//Incomplete or empty line is ignored
        tmpMaxTq.fEgrevo = atof(p);

        //Torque value
        p = strtok(NULL, " \n \t");
        if( p == NULL ) continue;//Incomplete or empty line is ignored
        tmpMaxTq.fEgtq = atof(p);

        p_TorquePointer->insert( tmpMaxTq );
    }
    fclose(fp_torquefile);

    //--------------------------------
    // If no data has been recorded
    //--------------------------------
    if(p_TorquePointer->empty()) return false;
```

```
    return true;
}

/**/
/*****************************************************************
 * Function name           : WriteSpecs
 * Function summary        :
 * Explanation             :
 *                         :
 * Argument (input)        : *fp : Analysis data output file name
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : true : Normal   false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool CFileIO::WriteSpecs(FILE *fp)
{
    int nRet;
    char   buf[1024];

    nRet = fprintf( fp, "VEHICLENAME \t%s \nTYPE \t%s \nSPEC FILE \t%s \nENGINE FILE \t%s \nTRANSMISSION FILE \t%s \nFINAL GEAR RATI
O \t%3.5f \nTIRE RADIUS(m) \t%3.5f \n ",
                    m_VehicleName.c_str(),
                    m_ConsDivision.c_str(),
                    m_SpecFileName.c_str(),
                    m_EngineFileName.c_str(),
                    m_TransmissionFileName.c_str(),
                    m_fLastReGear,
                    m_fTarR);

    if(nRet == EOF){
        cout << "(" << __LINE__ << ") ";
        cout << MSGWRITE_ERROR << endl;
        fclose(fp);
        return false;
    }

    fprintf( fp, " \n%s \tFC(km/l) \t%.4f \tAve.Speed(km/h) \t%5.1f \n", m_vModeNames[0].c_str(), m_AveFuel[0], m_AveSpeed[0]);
    fprintf( fp, "HIGHWAY \tFC(km/l) \t%.4f \tAve.Speed(km/h) \t%5.1f \n", m_AveFuel[1], m_AveSpeed[1]);
    fprintf( fp, "AVERAGE \tFC(km/l) \t%.4f \tHIGHWAY RATIO \t%1.2f \n", m_fFuelAverageCons, divValues[m_iConsDivisionIndex][5]
/100);
    fprintf( fp, "MID-TOWN \tFC(km/l) \t%.4f \tAve.Speed(km/h) \t%5.1f \n", m_AveFuel[2], m_AveSpeed[2]);

    return true;
}


/**/
/*****************************************************************
 * Function name           : WriteHead
 * Function summary        : Analysis data header output processing
 * Explanation             : Header is output to processing result file.
 *                         :
 * Argument (input)        : *fp : Analysis data output file name
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : true : Normal   false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool CFileIO::WriteHead(FILE *fp, int i)
{
    int    nRet;
    string specs;
    string szFieldTitle;
    double fAveFuel;

    nRet = fprintf( fp, "  \n%s \n", m_vModeNames[i-1].c_str() );
    if(nRet == EOF){
        cout << "(" << __LINE__ << ") ";
        cout << MSGWRITE_ERROR << endl;
        return( false );
    }

    szFieldTitle = DEF_PR_POS1;
    szFieldTitle = szFieldTitle + " \t" + DEF_PR_POS2;
    szFieldTitle = szFieldTitle + " \t" + DEF_PR_POS3;
    szFieldTitle = szFieldTitle + " \t" + DEF_PR_POS4;
    szFieldTitle = szFieldTitle + " \t" + DEF_PR_POS5;
    szFieldTitle = szFieldTitle + " \t" + DEF_PR_POS6;
    szFieldTitle = szFieldTitle + " \t" + DEF_PR_POS7;
    szFieldTitle = szFieldTitle + " \t" + DEF_PR_POS8;
    szFieldTitle = szFieldTitle + " \t" + DEF_PR_POS9;
```

```
    nRet = fprintf(fp, "%s \n", szFieldTitle.c_str());
    if (nRet == EOF){
        cout << "(" << __LINE__ << ") ";
        cout << MSGWRITE_ERROR << endl;
        return( false );
    }

    return true;
}


/**/
/*****************************************************************
 * Function name           : WriteValues
 * Function summary        :
 * Explanation             :
 *                         :
 * Argument (input)        : *fp : Analysis data output file name
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : true : Normal   false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool CFileIO::WriteValues(FILE *fp, int i)
{
    string szTmp;

    int  nRet;

    char buf[1024];

    bool tmpbN_norm_f=false;
    bool tmpbT_norm_f=false;
    bool tmpbTe_f = false;

    double fMaxTe;
    double tmpfVref;
    double tmpfVana;
    double tmpN_norm;
    double tmpT_norm;

    char   tmp_strfTe[128];
    char   tmp_strfNe[128];
    char   tmp_strN_norm[128];
    char   tmp_strT_norm[128];

    double tmpfDNe;
    double tmpfDTe;

    CCalculatePos *pCalPos;

    pCalPos = new CCalculatePos();

    for(p_mpData = mp_PtnData[i].begin() ; p_mpData != mp_PtnData[i].end() ; p_mpData++){

        fMaxTe = p_mpData->second.fMaxTe;

        tmpfDTe = p_mpData->second.fTe;
        tmpfDNe   = p_mpData->second.fNegrevo;


        tmpN_norm = (((p_mpData->second.fNegrevo - m_fIdleNe)/( m_fFixedNe - m_fIdleNe )) * 100.0);
        if(fMaxTe <= 0 ){
            tmpT_norm = 0;
        }
        else{
            tmpT_norm = (p_mpData->second.fTe*100.00) / fMaxTe;
        }


        if(tmpfDTe < 0 ){
            tmpbTe_f = true;
        }
        if(tmpN_norm<0){
            tmpbN_norm_f = true;
        }
        if(tmpT_norm < 0 ){
            tmpbT_norm_f = true;
        }

        tmpfVref = p_mpData->second.fVref_sp;
        tmpfVana = p_mpData->second.fVana_sp;
        tmpfDNe  = p_mpData->second.fNegrevo;

        if(m_bLossTqOut == false ){
            if(tmpbTe_f == false ){
```

```
                sprintf( tmp_strfTe, "%.1f", tmpfDTe );
            }
            else{
                sprintf( tmp_strfTe, "%s", "M" );
            }
        }
        else{
            sprintf( tmp_strfTe, "%.1f", tmpfDTe );
        }
        sprintf( tmp_strfNe, "%.1f", tmpfDNe );
        if(tmpbN_norm_f == false ){
            sprintf( tmp_strN_norm, "%.2f", tmpN_norm );
        }
        else{
            sprintf( tmp_strN_norm, "%s", "M" );
        }
        if(m_bLossTqOut == false ){
            if(tmpbT_norm_f == false ){
                sprintf( tmp_strT_norm, "%.2f", tmpT_norm );
            }
            else{
                sprintf( tmp_strT_norm, "%s", "M" );
            }
        }
        else{
            sprintf( tmp_strT_norm, "%.2f", tmpT_norm );
        }

        double fuel=p_mpData->second.fFuel;

        sprintf(buf,"%d \t%.2f \t%.2f \t%s \t%s \t%s \t%s \t%d \t%.6lf",
                (int)(p_mpData->second.fTimes),   // Accumulated time
                tmpfVref,              // Reference vehicle speed
                tmpfVana,              // Analysis vehicle speed
                tmp_strfNe,            // Engine speed
                tmp_strfTe,            // Engine torque
                tmp_strN_norm,
                tmp_strT_norm,
                p_mpData->second.nGear,           // Gear position
                fuel);

        nRet = fprintf(fp, "%s \n", buf);
        if (nRet == EOF){
            fclose(fp);
            cout << "(" << __LINE__ << ") ";
            cout << MSGWRITE_ERROR << endl;
            return false;
        }
    }

    delete (pCalPos);

    return true;
}

/**/
/*****************************************************************
 * Function name           : WriteAllCalculateData
 * Function summary        : Processed data output processing
 * Explanation             : Processing result is output to file.
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : true : Normal   false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool CFileIO::WriteCalculateData()
{
    bool   bRet;
    char   buf[1024];
    FILE   *m_pOutputFile;

    if((m_pOutputFile=fopen(m_OutputData.c_str(),"wt"))==NULL){
        cout << "(" << __LINE__ << ") ";
        sprintf( buf, "Error occured while creating %s file.",m_OutputData.c_str());
        cout << buf << endl;
        return false;
    }

    bRet = WriteSpecs(m_pOutputFile);
    if(!bRet) return false;


    if(m_bOutputResults){
        for(int i = 1; i <= NB_RIDING_MODE; i++ ){
```

```
            bRet = WriteHead(m_pOutputFile, i);
            if(!bRet) return false;

            bRet = WriteValues(m_pOutputFile, i);
            if(!bRet) return false;
        }
    }

    fclose(m_pOutputFile);
    return true;
}

CExecuteProc *pExecuteProc;

/**/
/*****************************************************************
 * Function name           : CCalculatePos
 * Function summary        : Constructor
 * Explanation             : Class constructor
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
CCalculatePos::CCalculatePos()
{
    return;
}
/**/
/*****************************************************************
 * Function name           : ~CCalculatePos
 * Function summary        : Destructor
 * Explanation             : Class destructor
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
CCalculatePos::~CCalculatePos()
{
    return;
}
/**/
/*****************************************************************
 * Function name           : CalcTqGear
 * Function summary        : Torque calculation modul
 * Explanation             : Torque is calculated. However, interpolation data is
 *                         : considered.
 * Argument (input)        : p1 : Calculate position
 * Argument (input)        : nGear : Gear position
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : true : Normal   false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool    CCalculatePos::CalcTqGear(map<double, stData>::iterator p1, int nGear )
{
    bool    bRet;
    double  fnGearPass;                 // n'th-grear ratio (transmission efficiency) data
    double  fTarR;                      // Tire rolling radius data
    double  fCarMt;                     // Vehicle body weight
    double  fGearRatio;
    double  fRL;
    double  fRS;
    double  fA;
    double  fTe;
    double  tmpTe;
    double  fMaxLossTq;
    double  fdf;

    double R;

    if( p1 == mp_PtnData[m_ModeID].end() ){
        return false;
    }

    if( nGear == 0 ){
        p1->second.fTe = 0.0;
```

```
        return true;
    }

    // Obtain gear transmission efficiency.
    fnGearPass = GetGearPass(nGear);

    bRet = GetGearFullRatio( nGear, fGearRatio);
    if( bRet == false ){
        return( false );
    }


    fRL = CalcRL(p1->second.fVana_sp);
    fRS = CalcRS(p1);

    R= fRL + fRS + (m_fCarWeight+GetCarRotWeight(nGear))/G * p1->second.fCarA;

    // Engine torque  Te = (Mt * Alfa / g + RL ) * (1000 * rd) / (Gti * Ut)
    fTe = ((G * m_fTarR) / fGearRatio) * R;

    if( R >= 0 ){       // +
        fTe /= (fnGearPass * UD);
    }else{              // -
        fTe *= (fnGearPass * UD);
    }

    p1->second.fTe = fTe;

    return true;
}

/**/
/*****************************************************************
 * Function name           : GetGearPass
 * Function summary        : Gear transmission efficiency setup processing
 * Explanation             : Gear transmission efficiency is set.
 *                         :
 * Argument (input)        : nGear : Gear position
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : double Transmission efficiency
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
double CCalculatePos::GetGearPass( int nGear )
{
    if( m_fGearRatio.empty() == true ){
        return( 0 );
    }

    if( nGear > (int)(m_fGearRatio.size()) ){
        return( 1 );
    }

    // ----------------------------
    // Set transmission efficiency based on gear ratio.
    // ----------------------------
    if(m_fSubGearRatio.empty() == true ){
        if( m_fGearRatio[nGear-1] == 1 ){               // If gear ratio is 1:0.
            return( DEF_FORCE_ON98 );
        }
        else{
            return( DEF_FORCE_OFF95 );
        }
    }
    else{
        nGear = nGear / m_fSubGearRatio.size() + ( nGear % m_fSubGearRatio.size() );
        if( m_fMainGearRatio[nGear-1] == 1 ){               // If gear ratio is 1:0.
            return( DEF_FORCE_ON98 );
        }
        else{
            return( DEF_FORCE_OFF95 );
        }
    }
}
/**/
/*****************************************************************
 * Function name           : CalcRL
 * Function summary        : Rolling resistance calculation processing
 * Explanation             : Rolling resistance is calculated.
 *                         :
 * Argument (input)        : fV : Vehicle speed
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : double Rolling resistance value
 * Created by              :
 * Updated on (created on) :
```

```
 * Remarks               :
 ****************************************************************/
double CCalculatePos::CalcRL(double fV)
{
    return( m_fMuRollRes * m_fCarWeight + m_fMuAir *(fV*fV) );
}

/**/
/****************************************************************
 * Function name           : CalcRS
 * Function summary        : Rolling resistance calculation processing
 * Explanation             : Rolling resistance is calculated.
 *                         :
 * Argument (input)        : fV : Vehicle speed
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : double Rolling resistance value
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 ****************************************************************/
double   CCalculatePos::CalcRS(map<double, stData>::iterator p1)
{
    return( m_fCarWeight * sin(atan( p1->second.fGrade / 100.0 )) );
}

/**/
/****************************************************************
 * Function name           : CalcNeGear
 * Function summary        : Engine speed calculation processing
 * Explanation             : Engine speed is calculated.
 *                         :
 * Argument (input)        : p1 : Calculate position
 * Argument (input)        : nGear : Gear position
 * Argument (input)        : iMode : 0:reference speed  1:calculated speed
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : true : Normal   false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 ****************************************************************/
bool    CCalculatePos::CalcNeGear(  map<double, stData>::iterator p1,
                                    int nGear, int iMode )
{
    double fV;
    double fNe;
    bool   bRet;
    double fGearti;
    double fTarR;                           // Tire rolling radius data
    map<double, stData>::iterator p_bef;

    p_bef = p1;
    if( p_bef != mp_PtnData[m_ModeID].begin() ){
        p_bef--;
    }
    if( p1 == mp_PtnData[m_ModeID].end() ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }

    if( nGear == 0 ){
        p1->second.fNegrevo = m_fIdleNe;
        return true;
    }

    if( iMode == 0 ){
        fV = p1->second.fVref_sp;
    }else{
        fV = p1->second.fVana_sp;
    }
    bRet = GetGearFullRatio( nGear, fGearti);
    if( bRet != true ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }

    fNe = fV / 60.0 * fGearti * 1000.0 / (2.0 * PI * m_fTarR);
    p1->second.fNegrevo = fNe;

    if( p_bef->second.fVana_sp == 0.0 && p1->second.fVref_sp != 0.0 ){
        if( iMode == 0 ){
            if( p1->second.fNegrevo < m_fCl_MetNe ){
                p1->second.fNegrevo = m_fCl_MetNe;
            }
        }
    }
```

```
    return true;
}
/**/
/*****************************************************************
 * Function name           : CalcVanaGear
 * Function summary        : Speed calculation processing
 * Explanation             : Speed is calculated.
 *                         :
 * Argument (input)        : p1 : Calculate position
 * Argument (input)        : nGear : Gear position
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : true : Normal   false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool  CCalculatePos::CalcVanaGear(  map<double, stData>::iterator p1, int nGear )
{
    double fV;
    double fNe;
    bool   bRet;
    double fGearRatio;
    map<double, stData>::iterator p_bef;
    double tmpfA;
    double tmpfVbef;

    p_bef = p1;
    if( p_bef != mp_PtnData[m_ModeID].begin() ){
        p_bef--;
    }
    if( p1 == mp_PtnData[m_ModeID].end() ){
        return false;
    }

    if( nGear == 0 ){
        p1->second.fVana_sp = p1->second.fVref_sp;
        return true;
    }

    if( p1->second.fNegrevo >= m_fOutRot ){
        p1->second.fNegrevo = m_fOutRot;
    }
    fNe = p1->second.fNegrevo;

    bRet = GetGearFullRatio( nGear, fGearRatio );
    if ( bRet != true ){
        return( false );
    }

    fV = fNe * 60.0 / fGearRatio / 1000.0 * (2.0 * PI * m_fTarR );
    if( fV >= p1->second.fVref_sp ){
        fV = p1->second.fVref_sp;
    }else if( fabs(fV - p1->second.fVref_sp) < 0.000000001 ){
        fV = p1->second.fVref_sp;
    }

    p1->second.fVana_sp = fV;
    p1->second.fCarA=((fV-p_bef->second.fVana_sp)/1.0)*1000.0/(60.0*60.0);// Calculates acceleration.
    p1->second.fA=((fV-p_bef->second.fVana_sp)/1.0);

    tmpfVbef = p_bef->second.fVana_sp;
    tmpfA = p1->second.fCarA;

    return( true );
}

/**/
/*****************************************************************
 * Function name           : GetCarRotWeight
 * Function summary        : Rotation partial equivalent mass data calculation processing
 * Explanation             : Rotation partial equivalent mass is calculated.
 *                         :
 * Argument (input)        : nGear : Gear position
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : double Curb vehicle weight value
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
double CCalculatePos::GetCarRotWeight(int nGear)
{
    double fW;
    double fGearRatio;

    GetGearRatio(nGear, fGearRatio);
```

```
    double fBeta;
    double fGearRatioCheck,fGearRatioCheckbef;

    if(( m_ConsDivision == "TT1" )||( m_ConsDivision == "T10" )||( m_ConsDivision == "T11" )){
        if(( m_nMaxSubGear == 1 )&&( m_nMaxMainGear == 7 )){
            GetGearRatio(m_nMaxMainGear, fGearRatioCheck);
            GetGearRatio(m_nMaxMainGear-1, fGearRatioCheckbef);
            if( fGearRatioCheck == 1.0 ){
                fBeta = OD_BETA_ST;
            }else if( fGearRatioCheckbef == 1.0 ){
                fBeta = OD_BETA_SOD;
            }else{
                fBeta = OD_BETA_DOD;
            }
        }else if(( m_nMaxSubGear == 2 )&&( m_nMaxMainGear == 6 )){
            GetGearRatio(m_nMaxGear, fGearRatioCheck);
            if( fGearRatioCheck == 1.0 ){
                fBeta = OD_BETA_ST;
            }else{
                fBeta = OD_BETA_DOD;
            }
        }else{
            fBeta = OD_BETA_DOD;
        }
    }else{
        fBeta = OD_BETA_DOD;
    }

    fW = (M_FACT + (E_FACT * (fGearRatio * fBeta) * (fGearRatio * fBeta))) * m_fCarCurbW;

    return (fW);
}
/**/
/*****************************************************************
 * Function name           : GetGearRatio
 * Function summary        : Gear ratio acquisition processing
 * Explanation             : Gear ratio is obtained from gear position.
 *                         :
 * Argument (input)        : nGear    : Gear position
 * Argument (output)       : fGearRatio : Gear ratio
 * Argument (I/O)          : None
 * Return value            : true : Normal   false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool CCalculatePos::GetGearRatio(int nGear,double &fGearRatio)
{
    if(m_fGearRatio.empty() == true ){              // If no data exists
        fGearRatio = 1;                             // Temporarily sets gear ratio to 1.
    return( false );
    }

    if(nGear > (int)(m_fGearRatio.size()) ){        // In case of larger than entered gear ratio
    fGearRatio = 1;                               // Temporarily sets gear ratio to 1.
    return( false );
    }

    fGearRatio = m_fGearRatio[nGear-1];             // Sets gear ratio.

    return( true );
}
/**/
/*****************************************************************
 * Function name           : GetGearFullRatio
 * Function summary        : Gear ratio acquisition (including final reduction ratio)
 * Explanation             : Gear ratio including final reduction ratio is obtained
 *                         : from gear position.
 * Argument (input)        : nGrear  : Gear position
 * Argument (output)       : fGearRatio : Gear ratio
 * Argument (I/O)          : None
 * Return value            : true : Normal   false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool CCalculatePos::GetGearFullRatio(int nGrear,double &fGearRatio)
{
    bool   bRet;
    double fnGearData;

    // n'th-gear ratio
    bRet = GetGearRatio( nGrear, fnGearData);
    if(bRet != true ){
        return( false );
    }

    fGearRatio = fnGearData * m_fLastReGear;
```

```
    return( true );
}
/**/
/*****************************************************************
 * Function name           : GetExF
 * Function summary        : Data setup processing for excess force ratio
 * Explanation             : Force ratio data is set.
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : true : Normal  false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool CCalculatePos::GetExF(void)
{
    stExF   tmpExF;                 // Temporary excess force ratio data
    bool    bRet;
    int     i;
    string  tmpStr;
    double  fGearRatio;
    int     ntmpMainGear;
    int     iMiniGear;

    // Existing excess force ratio data is provided.
    if( m_ExF.empty() != true ){            // If data already exists
        m_ExF.erase(m_ExF.begin(),
                    m_ExF.end() );      // Deletes existing data.
        m_ExF.clear();
    }

    if( m_nTorqueConv == 1 ){
        iMiniGear = 1;
    }else{
        iMiniGear = 2;
    }

    for( i = 1; i <= m_nMaxGear; i++ ){
        memset( &tmpExF, 0x00, sizeof( tmpExF ));   // Initializes temporary data.

        tmpExF.nGear = i;                           // Gear position

        // Gear position
        bRet = GetGearRatio( tmpExF.nGear, fGearRatio);
        if( bRet != true ){
            return( false );
        }
        tmpExF.fGearRatio = fGearRatio;             // Gear ratio
        // Transmission efficiency
        tmpExF.fForcePer = GetGearPass( i );        // Sets gear transmission efficiency.

        if( m_fSubGearRatio.empty() == true ){
            ntmpMainGear = i;
        }else{
            ntmpMainGear = i / m_fSubGearRatio.size() + ( i % m_fSubGearRatio.size() );
        }

        //-------------------------
        // Sets excess torque ratio
        //-------------------------
        if(m_fGrossVehicleWeight>=DEF_FOVER){
            if( ntmpMainGear <= iMiniGear )     tmpExF.fFreePer = DEF_F_OV_GEAR2;
            else if (ntmpMainGear == iMiniGear+1) tmpExF.fFreePer = DEF_F_OV_GEAR3;
            else if (ntmpMainGear >= iMiniGear+2) tmpExF.fFreePer = DEF_F_OV_GEAR4;
        }
        else{
            if( ntmpMainGear <= iMiniGear )     tmpExF.fFreePer = DEF_F_UN_GEAR2;
            else if (ntmpMainGear == iMiniGear+1) tmpExF.fFreePer = DEF_F_UN_GEAR3;
            else if (ntmpMainGear >= iMiniGear+2) tmpExF.fFreePer = DEF_F_UN_GEAR4;
        }

        //-------------------------
        // Sets normalized engine speed
        //-------------------------
        if( ntmpMainGear <= iMiniGear )    tmpExF.fMinPer  = DEF_F_NE_GEAR2;
        else if(ntmpMainGear == iMiniGear+1) tmpExF.fMinPer  = DEF_F_NE_GEAR3;
        else if(ntmpMainGear == iMiniGear+2) tmpExF.fMinPer  = DEF_F_NE_GEAR4;
        else if(ntmpMainGear >= iMiniGear+3) tmpExF.fMinPer  = DEF_F_NE_GEAR5;

        tmpExF.fMinNe = ( m_fFixedNe - m_fIdleNe ) * tmpExF.fMinPer/100.0 + m_fIdleNe;          // Calculates lower-limit
engine speed.
        m_ExF.insert( tmpExF );                         // Sets excess force ratio data.
    }

    return( true );
```

```
}
/**/
/*****************************************************************
 * Function name           : CalcAllData
 * Function summary        : Data setup processing for excess force ratio
 * Explanation             : Force ratio data is set.
 *                         :
 * Argument (input)        : pi : Calculate position
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : true : Normal  false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool    CCalculatePos::CalcAllData( map<double, stData>::iterator p1 )
{
    int bRet;
    double fMaxLossTq;

    if( p1->second.fVana_sp == 0.0 ){
        p1->second.fNegrevo = m_fIdleNe;
    }

    if( p1->second.nGear != 0 ){
        if( p1->second.fCarA > 0 ){
            if( p1->second.fNegrevo < m_fCl_MetNe ){
                p1->second.fNegrevo = m_fCl_MetNe;
            }
        }else{
            if( p1->second.fNegrevo < m_fCl_RelNe ){
                p1->second.fNegrevo = m_fIdleNe;
                p1->second.fTe = 0.0;
                p1->second.fMaxTe = 0.0;
                p1->second.nGear = 0;
            }
        }
    }else{
        if( p1->second.fNegrevo < m_fCl_RelNe ){
            p1->second.fNegrevo = m_fIdleNe;
            p1->second.fTe = 0.0;
            p1->second.fMaxTe = 0.0;
            p1->second.nGear = 0;
        }
    }

    if( p1->second.fNegrevo >= m_fOutRot ){
        bRet = CalcVanaGear( p1, p1->second.nGear );        // An analyzed car speed is calculated.
        if( bRet == false ){
            cout << "(" << __LINE__ << ") ";
            return false;
        }
        bRet = CalcTqGear( p1, p1->second.nGear );      // Engine torque (without complement)
        if( bRet == false ){
            cout << "(" << __LINE__ << ") ";
            return false;
        }
    }else{
        bRet = CalcTqGear( p1, p1->second.nGear );      // Engine torque (without complement)
        if( bRet == false ){
            cout << "(" << __LINE__ << ") ";
            return false;
        }
    }

    if( p1->second.fTe < 0 ){
        bRet = GetLRevMaxLossTq( p1->second.fNegrevo, fMaxLossTq );
        if( bRet == true ){
            if( p1->second.fTe < fMaxLossTq ){
                p1->second.bFuelCut = true;
            }
        }
    }

    p1->second.fMaxTe = GetLRevMaxTq( p1->second.fNegrevo );
    if( p1->second.bFuelCut == false ){
        if( p1->second.fNegrevo <= m_fIdleNe ){
            fuelSearch( -1, p1->second.fTe, p1->second.fFuel );
        }else{
            fuelSearch( p1->second.fNegrevo, p1->second.fTe, p1->second.fFuel );
        }
    }else{
        if( p1->second.nGear == 0 ){
            fuelSearch( -1, p1->second.fTe, p1->second.fFuel );
        }else{
            p1->second.fFuel = 0.0;
        }
    }
```

```
    return true;
}

/**/
/*************************************************************
 * Function name           : CalcTqGear
 * Function summary        : Torque calculation modul
 * Explanation             : Torque is calculated. However, interpolation data is
 *                         : considered.
 * Argument (input)        : p1 : Calculate position
 * Argument (input)        : nGear : Gear position
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : true : Normal   false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *************************************************************/
bool    CCalculatePos::CalcTqV( map<double, stData>::iterator p1,
                                int nGear )
{
    bool    bRet;
    double  fnGearPass;                 // n'th-grear ratio (transmission efficiency) data
    double  fTarR;                      // Tire rolling radius data
    double  fCarMt;                     // Vehicle body weight
    double  fGearRaito;
    double  fRL;
    double  fRS;
    double  fA;
    double  fTe;
    double  tmpTe;
    double  fMaxLossTq;
    double  fdf;
    double  fV;
    double  fNe;
    map<double, stData>::iterator p_bef;

    p_bef = p1;
    if( p_bef != mp_PtnData[m_ModeID].begin() ){
        p_bef--;
    }

    if( p1 == mp_PtnData[m_ModeID].end() ){
        return false;
    }

    if( nGear == 0 ){
        p1->second.fTe = 0.0;
        return true;
    }

    fTe = p1->second.fTe;

    // Vehicle body weight
    // fCarMt = W + ⊿W
    fCarMt = m_fCarWeight;
    fCarMt += GetCarRotWeight(nGear);

    // Obtain gear transmission efficiency.
    fnGearPass = GetGearPass(nGear);

    bRet = GetGearFullRatio( nGear, fGearRaito);
    if( bRet == false ){
        return( false );
    }

    fRL = CalcRL(p1->second.fVana_sp);
    fRS = CalcRS(p1);
    fRL = fRL + fRS;

    // + controle
    //          Te = ((9.8 * r) / (i(m) * i(f))) *
    //               ((μ(r) * W) + (s * W / 100.0) + (μ(α) * A * V(t)^2) + ((W + ⊿W) / 9.8 * (V(t) - V(t - 1)) / 3.6))
 / (η(m) * η(f))
    // - controle
    //          Te = ((9.8 * r) / (i(m) * i(f))) *
    //               ((μ(r) * W) + (s * W / 100.0) + (μ(α) * A * V(t)^2) + ((W + ⊿W) / 9.8 * (V(t) - V(t - 1)) / 3.6))
 * (η(m) * η(f))
    fA = ((fTe * (fnGearPass * UD) ) / ( (G * m_fTarR) / fGearRaito) - fRL ) * G / fCarMt;


    p1->second.fCarA = fA;

    fV = ((fA * (60.0*60.0)/ 1000.0) * 1.0) + p_bef->second.fVana_sp;                           // Calculates acceleration.
    p1->second.fVana_sp = fV;

    fNe = fV / 60.0 * fGearRaito * 1000.0 / (2.0 * PI * m_fTarR);
```

```
    p1->second.fNegrevo = fNe;

    return true;
}

/*************************************************************
 * Function name           : CalcMaxSp
 * Function summary        : Target-speed follow calculation processing
 * Explanation             : When speed changes from A to B,
 *                         : speed fV is calculated if target-speed follow is impossible in time fTm.
 * Argument (input)        : p1 : Calculate position
 * Argument (input)        : nGear : Gear position
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : true : Converged   false : Not converged
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *************************************************************/
bool    CCalculatePos::CalcMaxSp(   map<double, stData>::iterator p1,
                                    int nGear, int iMode)
{
    double                          fV_cal1;    // Calculation point intermediate-1
    double                          fV_calM;    // Calculation point intermediate
    double                          fV_cal2;    // Calculation point intermediate+1
    double                          fTe_DIF_1;  // Torque ratio of calculation point intermediate-1
    double                          fTe_DIF_M;  // Torque ratio of calculation point intermediate
    double                          fTe_DIF_2;  // Torque ratio of calculation point intermediate+1
    double                          fNe_def;    // Engine speed for this time (no correction)
    double                          fNe_cal;    // Engine speed for this time (after calculation)
    double                          fTe_def;    // Torque for this time (no correction)
    double                          fTe_spl;    // Torque for this time (corrected)
    double                          fTe_cal;    // Torque for this time (after calculation)
    double                          fCarA;      // Acceleration
    bool                            bRet;
    int                             nRet;       // Function return value
    int                             i;
    double                          fV;
    double                          fNe;
    map<double, stData>::iterator   p_bef;
    char wkstr[256];

    p_bef = p1;
    if( p_bef != mp_PtnData[m_ModeID].begin() ){
        p_bef--;
    }

    fV = p1->second.fVref_sp;

    // Calculate acceleration, torque, and engine speed for this time.
    bRet = CalcNeGear( p1, nGear );     // Engine speed
    if( bRet != true ){
        return false;
    }

    if( p1->second.fNegrevo >= m_fOutRot ){
        bRet = CalcVanaGear( p1, nGear );       // An analyzed car speed is calculated.
        if( bRet != true ){
            return( false );
        }
    }else{
        bRet = CalcVanaGear( p1, nGear );       // An analyzed car speed is calculated.
        if( bRet != true ){
            return( false );
        }
    }

    if( iMode != 0 ){
        if( p1->second.fNegrevo < m_fCl_MetNe ){
            p1->second.fNegrevo = m_fCl_MetNe;
        }
    }


    fNe = p1->second.fNegrevo;
    fNe_def = fNe;

    bRet = CalcTqGear( p1, nGear );
    if( bRet == false ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }
    fTe_def = p1->second.fTe;
    fTe_spl = GetLRevMaxTq(fNe_def);

    if( p1->second.fCarA < 0 ){
        return( true );
    }
```

```
if( fTe_spl < fTe_def ){                    // If spline-complemented torque is
    bRet = CalcNeGear( p1, nGear );         // Engine speed
    if( bRet == false ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }
    fNe_def = p1->second.fNegrevo;
    if( fNe_def < m_fCl_MetNe ){
        fNe_def = m_fCl_MetNe;
    }
    bRet = CalcTqGear( p1, nGear );         // Engine torque (without complement)
    if( bRet == false ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }
                                            // smaller than calculated torque
    fTe_cal = p1->second.fTe;               // Torque to be calculated
    fNe_cal = p1->second.fNegrevo;          // Engine speed to be calculated

    for( i=0; i<10000; i++ ){               // Finds approximate TeMax=Te and acceleration.
        // --------------------------------
        // Calculate intermediate speed.
        // --------------------------------
        fTe_cal = fTe_spl + (fTe_def - fTe_spl)/2.0;            // Determines mid-point speed.
        p1->second.fTe      = fTe_cal;
        bRet = CalcTqV( p1, nGear );        // An analyzed car speed is calculated.
        if( bRet != true ){
            return( false );
        }

        fNe_cal = p1->second.fNegrevo;
        fTe_spl = GetLRevMaxTq(fNe_cal);        // Linear interpolation
        fV = p1->second.fVana_sp;
        if( fNe_cal >= m_fOutRot ){
            p1->second.fNegrevo = m_fOutRot;
            fNe_cal = p1->second.fNegrevo;
            bRet = CalcVanaGear( p1, nGear );   // An analyzed car speed is calculated.
            if( bRet == false ){
                cout << "(" << __LINE__ << ") ";
                return( false );
            }

            bRet = CalcTqGear( p1, nGear );         // Engine torque (without complement)
            if( bRet == false ){
                cout << "(" << __LINE__ << ") ";
                return false;
            }
            fTe_def = p1->second.fTe;
        }

        if( fNe_cal < m_fCl_MetNe ){
            p1->second.fNegrevo = m_fCl_MetNe;
            fNe_cal = p1->second.fNegrevo;

            fTe_spl = GetLRevMaxTq(fNe_cal);        // Linear interpolation
            p1->second.fTe = fTe_spl;
            bRet = CalcTqV( p1, nGear );        // An analyzed car speed is calculated.
            if( bRet == false ){
                cout << "(" << __LINE__ << ") ";
                return false;
            }

            fV = p1->second.fVana_sp;

            sprintf( wkstr, "## Warning 5%% lower Max trq.##( %.f )", p1->second.fTimes );
            pExecuteProc->pConsole->myWriteConsole(3, 18, wkstr);
            break;
        }

        if( fTe_spl != 0.0 ){
            if( fabs( fTe_spl - fTe_cal  ) < 1.0E-6 ){  // If nearest value is found
                 break;
            }
        }

        bRet = CalcTqGear( p1, nGear );
        if( bRet == false ){
            cout << "(" << __LINE__ << ") ";
            return false;
        }
        fTe_def = p1->second.fTe;
        fTe_spl = GetLRevMaxTq(fNe_cal);

    }

    p1->second.fVana_sp = fV;
    if( i >= 10000 ){
```

```
            sprintf( wkstr, "## Warning 5%% lower Max trq.##( %.f )", p1->second.fTimes );
            pExecuteProc->pConsole->myWriteConsole(3, 18, wkstr);
        }
        bRet = CalcNeGear( p1, nGear, 1 );              // Engine speed
        if( bRet == false ){
            cout << "(" << __LINE__ << ") ";
            return false;
        }

        if( iMode != 0 ){
            if( p1->second.fNegrevo < m_fCl_MetNe ){
                p1->second.fNegrevo = m_fCl_MetNe;
            }
        }
    }

    p1->second.fVana_sp = fV;
    p1->second.fCarA = (( fV - p_bef->second.fVana_sp) / 1.0) * 1000.0/(60.0*60.0); // Calculates acceleration.
    p1->second.fA = (( fV - p_bef->second.fVana_sp) / 1.0);

    return( true );
}
/**/
/*****************************************************************
 * Function name           : GetLRevMaxTq
 * Function summary        : Max. torque data interpolation processing
 * Explanation             : Obtain max. torque data, and execute calculation. (Linear interpolation)
 *                         :
 * Argument (input)        : fNe : Engine speed
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : double Torque
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
double  CCalculatePos::GetLRevMaxTq( double fNe )
{
    double  fNeA,fNeB,fTq1,fTq2,fMaxTq;
    stMaxTq  tmpMaxTq;                      // Temporary max. torque data

    // Check max. torque data within data range if it exists.
    if (m_MaxTq.empty() != true ){
    }else{
        // Return NG if max. torque data does not exist.
        return( 0.0 );
    }

    // Find appropriate engine speed and max. loss torque data from array.
    memset( &tmpMaxTq, 0x00, sizeof( tmpMaxTq ));
    tmpMaxTq.fEgrevo = fNe;
    p_MaxTq = m_MaxTq.lower_bound( tmpMaxTq );

    // Return NG if not found.
    if( p_MaxTq == m_MaxTq.end() ){
        p_MaxTq = m_MaxTq.end();
        p_MaxTq--;
        fNeB = p_MaxTq->fEgrevo;
        fTq2 = p_MaxTq->fEgtq;
        p_MaxTq--;
        fTq1 = p_MaxTq->fEgtq;
        fNeA = p_MaxTq->fEgrevo;
        // Prevent dividing by 0.
        if ((fNeB - fNeA) == 0) return( 0.0 );
        // Obtain appropriate max. torque by linear interpolation (polygonal line).
        fMaxTq = fTq2 +
            (fTq2 - fTq1) /
            (fNeB - fNeA) *
            (fNe - fNeB);
        return( fMaxTq );
    }

    fNeB = p_MaxTq->fEgrevo;
    fTq2 = p_MaxTq->fEgtq;
    // Obtain preceding data.
    if( p_MaxTq != m_MaxTq.begin() ){
        p_MaxTq--;
        fTq1 = p_MaxTq->fEgtq;
        fNeA = p_MaxTq->fEgrevo;
    }else{                                  // Next data if first data.
        fTq1 = p_MaxTq->fEgtq;
        fNeA = p_MaxTq->fEgrevo;
        p_MaxTq++;
        fNeB = p_MaxTq->fEgrevo;
        fTq2 = p_MaxTq->fEgtq;
    }

    // Prevent dividing by 0.
```

```
    if ((fNeB - fNeA) == 0) return( 0.0 );

    // Obtain appropriate max. torque by linear interpolation (polygonal line).
    fMaxTq = fTq1 +
            (fTq2 - fTq1) /
            (fNeB - fNeA) *
            (fNe - fNeA);

    return( fMaxTq );
}

/**/
/*****************************************************************
 * Function name           : GetLRevMaxLossTq
 * Function summary        : Max. loss-torque data interpolation processing
 * Explanation             : Obtain max. loss-torque data, and execute calculation. (Linear interpolation)
 *                         :
 * Argument (input)        : fNe : Engine speed
 * Argument (output)       : double fMaxLossTorque
 * Argument (I/O)          : None
 * Return value            : true : Normal  false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool    CCalculatePos::GetLRevMaxLossTq(double fNe, double &fMaxLossToque)
{
    double  fNeA,fNeB,fTq1,fTq2,fMaxTq;
    stMaxTq  tmpMaxTq;                      // Temporary max. torque data

    // Check max. torque data within data range if it exists.
    if (m_MaxLossTq.empty() != true ){
    }else{
        // Return NG if max. torque data does not exist.
        return( 0.0 );
    }

    // Find appropriate engine speed and max. loss torque data from array.
    memset( &tmpMaxTq, 0x00, sizeof( tmpMaxTq ));
    tmpMaxTq.fEgrevo = fNe;
    p_MaxTq = m_MaxLossTq.lower_bound( tmpMaxTq );

    // Return NG if not found.
    if( p_MaxTq == m_MaxLossTq.end() ){
        p_MaxTq = m_MaxLossTq.end();
        p_MaxTq--;
        fNeB = p_MaxTq->fEgrevo;
        fTq2 = p_MaxTq->fEgtq;
        p_MaxTq--;
        fTq1 = p_MaxTq->fEgtq;
        fNeA = p_MaxTq->fEgrevo;
        // Prevent dividing by 0.
        if ((fNeB - fNeA) == 0) return( 0.0 );
        // Obtain appropriate max. torque by linear interpolation (polygonal line).
        fMaxLossToque = fTq2 +
            (fTq2 - fTq1) /
            (fNeB - fNeA) *
            (fNe - fNeB);
        return( true );
    }

    fNeB = p_MaxTq->fEgrevo;
    fTq2 = p_MaxTq->fEgtq;
    // Obtain preceding data.
    if( p_MaxTq != m_MaxLossTq.begin() ){
        p_MaxTq--;
        fTq1 = p_MaxTq->fEgtq;
        fNeA = p_MaxTq->fEgrevo;
    }else{                                  // Next data if first data.
        fTq1 = p_MaxTq->fEgtq;
        fNeA = p_MaxTq->fEgrevo;
        p_MaxTq++;
        fNeB = p_MaxTq->fEgrevo;
        fTq2 = p_MaxTq->fEgtq;
    }

    // Prevent dividing by 0.
    if ((fNeB - fNeA) == 0) return( 0.0 );

    // Obtain appropriate max. torque by linear interpolation (polygonal line).
    fMaxLossToque = fTq1 +
            (fTq2 - fTq1) /
            (fNeB - fNeA) *
            (fNe - fNeA);

    return true;
}
/**/
```

```
/*****************************************************************
 * Function name           : CForwardCalc
 * Function summary        : Constructor
 * Explanation             : Class constructor
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
CForwardCalc::CForwardCalc()
{
    pCalPos = new CCalculatePos;
    return;
}
/**/
/*****************************************************************
 * Function name           : ~CForwardCalc
 * Function summary        : Destructor
 * Explanation             : Class destructor
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
CForwardCalc::~CForwardCalc()
{
    delete pCalPos;
    return;
}
/**/
/*****************************************************************
 * Function name           : CalcGearCheck
 * Function summary        : Calculate gear setting main module
 * Explanation             : Point reading does a gear until the gear holding time.
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : p1 : Calculate position
 * Return value            : true : Normal   false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool    CForwardCalc::CalcGearCheck(map<double, stData>::iterator &p1 )
{
    int i;
    int tmpGear;
    int nSetGear;
    bool bRet;
    int iMode;
    int iRet;
    map<double, stData>::iterator p_before;
    map<double, stData>::iterator p_tmp;
    map<double, stData>::iterator p_chkp;

    p_chkp = p1;
    p_before = p1;
    if( p_before != mp_PtnData[m_ModeID].begin() ){
        p_before--;
    }

    if( mp_PtnData[m_ModeID].empty() ){
        cout << "data empty.(" << __LINE__ << ") ";
        return false;
    }

    // Starting?
    if( p_before->second.fVana_sp == 0.0 && p1->second.fVref_sp != 0.0 ){
        bRet = CalcStFollow( p1 );
        if( bRet != true ){
            cout << "(" << __LINE__ << ") ";
            return false;
        }
        if( p1->second.fNegrevo < m_fCl_MetNe ){
            p1->second.fNegrevo = m_fCl_MetNe;
        }

        return true;
    }
```

```
// Point reading for three seconds is computed as to the action.
for( i = 0; i<1; i++ ){
    if( p1 == mp_PtnData[m_ModeID].end() ){
        break;
    }
    tmpGear = p_before->second.nGear;

    // Idle ?
    if( p_before->second.fVana_sp == 0.0 && p1->second.fVref_sp == 0.0 ){
        tmpGear = 0;
        p1->second.nGearTime = 3;
        continue;
    }

    if( tmpGear == 0 && p_before->second.fVana_sp != 0.0 && p1->second.fVref_sp != 0.0 ){
        p_tmp = p_before;
        while(1){
            if( p_tmp == mp_PtnData[m_ModeID].begin() ){
                break;
            }
            if( p_tmp->second.nGear != 0 ){
                tmpGear = p_tmp->second.nGear;
                break;
            }
            p_tmp--;
        }
    }

    if( p_before->second.fVana_sp > p1->second.fVref_sp ){
        p1->second.nGear = tmpGear;
        p1->second.nGearTime = p_before->second.nGearTime +1;
        // spped down?
    }else{
        nSetGear = tmpGear;

        bRet = CheckNowShiftChange( p1, nSetGear, tmpGear, 4 );
        if( bRet == false ){
            bRet = CalcGear( p1, nSetGear, tmpGear, 3 );            // Calculate gear setting module
        }else{
            bRet = CalcGear( p1, nSetGear, tmpGear, 0 );            // Calculate gear setting module
        }

        if( bRet != true ){
            cout << "(" << __LINE__ << ") ";
            p1 = p_chkp;
            return false;
        }
        p1->second.nGear = nSetGear;

        // Does a gear change?
        if( tmpGear != nSetGear ){
            p1->second.nGearTime = 1;
            p1->second.nGear = nSetGear;
            tmpGear = nSetGear;
        }else{
            p1->second.nGearTime = p_before->second.nGearTime +1;
        }
    }

    // Calculate acceleration, torque, and engine speed for this time.
    bRet = pCalPos->CalcNeGear( p1, tmpGear );      // Engine speed
    if( bRet != true ){
        cout << "(" << __LINE__ << ") ";
        p1 = p_chkp;
        return false;
    }

    bRet = pCalPos->CalcVanaGear( p1, tmpGear );    // An analyzed car speed is calculated.
    if( bRet != true ){
        cout << "(" << __LINE__ << ") ";
        p1 = p_chkp;
        return false;
    }

    if( tmpGear != 0 ){
        bRet = pCalPos->CalcMaxSp( p1, tmpGear );   // Target-speed follow calculation.
        if( bRet == false ){
            cout << "(" << __LINE__ << ") ";
            p1 = p_chkp;
            return false;
        }
    }

    p_before++;
    p1++;
}

p1 = p_chkp;
```

```
    return true;
}
/**/
/*****************************************************************
 * Function name           : CheckNowShiftChange
 * Function summary        : Calculate gear Force check
 * Explanation             :
 *                         :
 * Argument (input)        : p1         : Calculate start position
 * Argument (input)        : nGrear     : Gear position (calculate object)
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : true : Shift-Change OK   false : NG
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool    CForwardCalc::CheckNowShiftChange( map<double, stData>::iterator p1,
                            int nGear, int nOrgGear, int iMode)
{
    int tmpGear;
    stExF tmpExF;                               // Structure for search
    double  fNeMinLimit;                        // Engine speed determination lower limit
    double  fNeMaxTopLimit;                     // Max. engine speed rate
    double  fMaxLossTq;
    map<double, stData>::iterator p_before;
    bool bRet;

    p_before = p1;
    if( p_before != mp_PtnData[m_ModeID].begin() ){
        p_before--;
    }

    tmpGear = nGear;
    memset( &tmpExF, 0x00, sizeof( tmpExF ));   // Initializes search structure.
    tmpExF.nGear = nGear;                       // Sets gear.
    p_ExF = m_ExF.find( tmpExF );               // Obtains applicable gear position.
    fNeMinLimit = p_ExF->fMinNe;                // Sets lower-limit engine speed.
    fNeMaxTopLimit = m_fOutRot;

    // Calculate acceleration, torque, and engine speed for this time.
    bRet = pCalPos->CalcNeGear( p1, tmpGear );      // Engine speed
    if( bRet != true ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }

    bRet = pCalPos->CalcVanaGear( p1, tmpGear );    // An analyzed car speed is calculated.
    if( bRet == false ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }

    bRet = pCalPos->CalcTqGear( p1, tmpGear );      // Engine torque (without complement)
    if( bRet == false ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }

    p1->second.fMaxTe = pCalPos->GetLRevMaxTq( p1->second.fNegrevo );

    if( p1->second.fNegrevo <= fNeMinLimit ){
        return false;
    }
    if( p1->second.fNegrevo >= fNeMaxTopLimit ){
        return false;
    }

    if( iMode == 4 ){
        return true;
    }

    if( p1->second.fCarA < 0 ){
         return false;
    }

    if( p_ExF->fFreePer < p1->second.fMaxTe / p1->second.fTe ){
        return true;
    }else{
        if( iMode != 0 ){
            if( iMode == 2 ){
                if( p1->second.fCarA >= 0 ){
                    return true;
                }
            }
            return false;
        }else{
```

```
            if( p1->second.fTe > 0 ){
                return false;
            }else{
                return true;
            }
        }
    }

}
/**/
/*****************************************************************
 * Function name           : CheckNTimesHold
 * Function summary        : Calculate n sec hold check
 * Explanation             :
 *                         :
 * Argument (input)        : p1         : Calculate start position
 * Argument (input)        : p2         : Calculate end   position
 * Argument (input)        : nGrear     : Gear position (calculate object)
 * Argument (input)        : nOrgGrear  : Gear position (Now object)
 * Argument (output)       : fDiffV     : diff. Speed
 * Argument (I/O)          : None
 * Return value            : true : Shift-Change OK   false : NG
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
int CForwardCalc::CheckNTimesHold( map<double, stData>::iterator p1,
                                   map<double, stData>::iterator p2,
                                   int nGear, int nOrgGear, double &fDiffV, int iMode)
{
    map<double,stData>::iterator p_tmpData;     // Temporary pointer
    map<double,stData>::iterator p_befData;     // Temporary pointer
    map<double,stData>::iterator p_p1befData;   // Temporary pointer
    double  fNeMinLimit;                        // Engine speed determination lower limit
    double  fNeMaxTopLimit;                     // Max. engine speed rate
    stExF tmpExF;                               // Structure for search
    bool bRet;
    int  iRet;

    memset( &tmpExF, 0x00, sizeof( tmpExF ));   // Initializes search structure.
    tmpExF.nGear = nGear;                       // Sets gear.
    p_ExF = m_ExF.find( tmpExF );               // Obtains applicable gear position.
    fNeMinLimit = p_ExF->fMinNe;                // Sets lower-limit engine speed.
    fNeMaxTopLimit = m_fOutRot;

    p_p1befData = p1;
    if( p1 != mp_PtnData[m_ModeID].begin() ){
        p_p1befData--;                                  // Preceding speed
    }

    for( p_tmpData = p1; p_tmpData != p2; p_tmpData++){ // All sections subject to determination
        if( p_tmpData == mp_PtnData[m_ModeID].end() ){
            break;
        }
        p_befData = p_tmpData;
        if( p_tmpData != mp_PtnData[m_ModeID].begin() ){
            p_befData--;                                    // Preceding speed
        }

        // Calculate acceleration, torque, and engine speed for this time.
        bRet = pCalPos->CalcNeGear( p_tmpData, nGear );     // Engine speed
        if( bRet == false ){
            cout << "(" << __LINE__ << ") ";
            return -99;
        }
        bRet = pCalPos->CalcVanaGear( p_tmpData, nGear );   // An analyzed car speed is calculated.
        if( bRet == false ){
            cout << "(" << __LINE__ << ") ";
            return -99;
        }
        p_tmpData->second.fCarA =
                            ( p_tmpData->second.fVref_sp - p_befData->second.fVana_sp ) *
                            1000.0/(60.0*60.0);
        bRet = pCalPos->CalcTqGear( p_tmpData, nGear );     // Engine torque (without complement)
        if( bRet == false ){
            cout << "(" << __LINE__ << ") ";
            return -99;
        }

        p_tmpData->second.fMaxTe = pCalPos->GetLRevMaxTq( p_tmpData->second.fNegrevo );

        if( p_tmpData->second.fCarA < 0 ){
            continue;
        }

        if( p_tmpData->second.fNegrevo >= fNeMaxTopLimit){  // If engine speed is equal to or
            return -1;
        }
```

```
        if( p_tmpData->second.fNegrevo <=
                                        fNeMinLimit){       // In case of min. engine speed or less
            return -2;
        }

    }

    iRet = 0;
    fDiffV = 0.0;
    for( p_tmpData = p1; p_tmpData != p2; p_tmpData++){ // All sections subject to determination
        if( p_tmpData == mp_PtnData[m_ModeID].end() ){
            break;
        }

        p_befData = p_tmpData;
        if( p_tmpData != mp_PtnData[m_ModeID].begin() ){
            p_befData--;                                    // Preceding speed
        }

        bRet = pCalPos->CalcVanaGear( p_tmpData, nGear );   // An analyzed car speed is calculated.
        if( bRet == false ){
            cout << "(" << __LINE__ << ") ";
            return -99;
        }
        p_tmpData->second.fCarA =
                            ( p_tmpData->second.fVref_sp - p_befData->second.fVana_sp ) *
                            1000.0/(60.0*60.0);
        bRet = pCalPos->CalcTqGear( p_tmpData, nGear );     // Engine torque (without complement)
        if( bRet == false ){
            cout << "(" << __LINE__ << ") ";
            return -99;
        }

        p_tmpData->second.fMaxTe = pCalPos->GetLRevMaxTq( p_tmpData->second.fNegrevo );

        if( iMode != 2 ){
            if( p_befData->second.fVana_sp > p_tmpData->second.fVana_sp   ){
                break;
            }
        }

        bRet = pCalPos->CalcMaxSp( p_tmpData,
                                nGear );                // Target-speed follow calculation processing
        if( bRet == false ){
                cout << "(" << __LINE__ << ") ";
                return -99;
        }

        if( p_tmpData->second.fNegrevo >= fNeMaxTopLimit){  // If engine speed is equal to or
            return -1;
        }
        if( p_tmpData->second.fNegrevo <=
                                        fNeMinLimit){       // In case of min. engine speed or less
            return -2;
        }

        fDiffV = fDiffV +
                    fabs(p_tmpData->second.fVana_sp - p_tmpData->second.fVref_sp );
    }

    return iRet;

}
/**/
/*****************************************************************
 * Function name           : Check3TimesHold
 * Function summary        : Calculate 3 sec hold check
 * Explanation             :
 *                         :
 * Argument (input)        : p1         : Calculate start position
 * Argument (input)        : nGrear     : Gear position (calculate object)
 * Argument (input)        : nOrgGrear  : Gear position (Now object)
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : true : Shift-Change OK   false : NG
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool    CForwardCalc::Check3TimesHold( map<double, stData>::iterator p1,
                                int nGear, int nOrgGear, int iMode)
{
    stData tmpData;                             // Temporary analysis data
    map<double,stData>::iterator p2;
    int iRet;
    double fDiffV;

    // Search for prospective gear for conditions.
```

```
    memset( &tmpData, 0x00, sizeof( tmpData )); // Initializes temporary data.
    tmpData.fTimes = p1->second.fTimes + GEAR_HOLD_TIME;        // Sets gear hold time for shift-up.

    p2 = mp_PtnData[m_ModeID].find( tmpData.fTimes );           // Up to end of gear hold time

    iRet = CheckNTimesHold( p1, p2, nGear, nOrgGear, fDiffV, iMode );
    if( iRet == 0 ){
        if(( iMode == 0 )&&( fDiffV != 0.0 )){
            return false;
        }
        return true;
    }else{
        return false;
    }
}
/**/
/*****************************************************************
 * Function name           : CheckNTimesSpeedHold
 * Function summary        : Calculate 3 sec hold check
 * Explanation             :
 *                         :
 * Argument (input)        : p1         : Calculate start position
 * Argument (input)        : p2         : Calculate end   position
 * Argument (input)        : nGrear     : Gear position (calculate object)
 * Argument (input)        : nOrgGear   : Gear position (Now object)
 * Argument (output)       : nSetGear   : Gear position (setting gear)
 * Argument (I/O)          : None
 * Return value            : true : Shift-Change OK   false : NG
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool    CForwardCalc::CheckNTimesSpeedHold( map<double, stData>::iterator p1,
                            map<double, stData>::iterator p2,
                            int nGear, int nOrgGear, int &nSetGear, int iMode)
{
    stData tmpData;                             // Temporary analysis data
    int tmpGear;
    int tmpMaxGear;
    int nOffsetGear;
    int iRet;
    int iMiniGear;
    double fDiffV;
    double ftmpDiffV;
    multimap<int,double> mDiff;                     // diffrent reference speed data
    multimap<int,double>::iterator p_Diff;          // diffrent reference spped data pointer
    map<double,stData>::iterator tmp_p1;
    map<double,stData>::iterator tmp_p2;
    bool bRet;


    // The gear before raising a shift is held.
    tmpGear = nGear;
    fDiffV = 0;
    nSetGear = 0;
    nOffsetGear = 3;
    if( m_fSubGearRatio.empty() != true ){
        nOffsetGear = nOffsetGear * m_fSubGearRatio.size();
    }
    if( tmpGear + nOffsetGear > m_nMaxGear ){
        tmpMaxGear = m_nMaxGear;
    }else{
        tmpMaxGear = tmpGear +nOffsetGear;
    }

    iMiniGear = m_nInitGear;
    if( m_fSubGearRatio.size() >= 2 ){
        if(( iMode == 3 )){
            iMiniGear = iMiniGear - m_fSubGearRatio.size() + 1;
        }
    }

    if( iMode == 0 ){
        if( p1 != p2 ){
            // first. Now Gear hold and next time shift change.
            for( tmp_p1 = p1, tmp_p2 = p1; tmp_p1 != p2; tmp_p1++ ){
                tmp_p2++;
                ftmpDiffV = 0.0;
                iRet = CheckNTimesHold( p1, tmp_p2, nOrgGear, nOrgGear, ftmpDiffV, 0 );
                if( iRet == -3 || iRet == 0 ){
                    for(;tmpGear <= tmpMaxGear ;tmpGear++){
                        bRet = CheckNowShiftChange( tmp_p2, tmpGear, nOrgGear, 0 );
                        if(( bRet == true )||( tmpGear == nOrgGear )){
                            iRet = CheckNTimesHold( tmp_p2, p2, tmpGear, nOrgGear, fDiffV, 0 );
                            if( iRet == -3 || iRet == 0 ){
                                fDiffV = fDiffV + ftmpDiffV;
                                mDiff.insert(pair<int,double>(nOrgGear, fDiffV) );
                            }
```

```
                    }
                }
            }
        }
    }

    for(tmpGear = nOrgGear;tmpGear <= tmpMaxGear ;tmpGear++){
        iRet = CheckNTimesHold( p1, p2, tmpGear, nOrgGear, fDiffV, 0 );
        if( iRet == -3 || iRet == 0 ){
            mDiff.insert(pair<int,double>(tmpGear, fDiffV) );
        }
    }

}else if(( iMode == 1 )||( iMode == 3 )){
    if( p1 != p2 ){
        // first. Now Gear hold and next time shift change.
        for( tmp_p1 = p1, tmp_p2 = p1; tmp_p1 != p2; tmp_p1++ ){
            tmp_p2++;
            ftmpDiffV = 0.0;
            iRet = CheckNTimesHold( p1, tmp_p2, nOrgGear, nOrgGear, ftmpDiffV, 1 );
            if( iRet == -3 || iRet == 0 ){
                for(tmpGear = nOrgGear; tmpGear >= iMiniGear ;tmpGear--){
                    bRet = CheckNowShiftChange( tmp_p2, tmpGear, nOrgGear, 0 );
                    if(( bRet == true )||( tmpGear == nOrgGear )||( iMode == 3 )){
                        iRet = CheckNTimesHold( tmp_p2, p2, tmpGear, nOrgGear, fDiffV, 1 );
                        if( iRet == -3 || iRet == 0 ){
                            fDiffV = fDiffV + ftmpDiffV;
                            mDiff.insert(pair<int,double>(nOrgGear, fDiffV) );
                        }
                    }
                }
            }
        }
    }

    for(tmpGear = nOrgGear; tmpGear >= iMiniGear ;tmpGear--){
        iRet = CheckNTimesHold( p1, p2, tmpGear, nOrgGear, fDiffV, 1 );
        if( iRet == -3 || iRet == 0 ){
            mDiff.insert(pair<int,double>(tmpGear, fDiffV) );
        }
    }
}else if( iMode == 2 ){
    if( p1 != p2 ){
        // first. Now Gear hold and next time shift change.
        for( tmp_p1 = p1, tmp_p2 = p1; tmp_p1 != p2; tmp_p1++ ){
            tmp_p2++;
            ftmpDiffV = 0.0;
            iRet = CheckNTimesHold( p1, tmp_p2, nOrgGear, nOrgGear, ftmpDiffV, 1 );
            if( iRet == -3 || iRet == 0 ){
                for(tmpGear = tmpMaxGear; tmpGear >= nOrgGear ;tmpGear--){
                    bRet = CheckNowShiftChange( tmp_p2, tmpGear, nOrgGear, 0 );
                    if(( bRet == true )||( tmpGear == nOrgGear )){
                        iRet = CheckNTimesHold( tmp_p2, p2, tmpGear, nOrgGear, fDiffV, 1 );
                        if( iRet == -3 || iRet == 0 ){
                            fDiffV = fDiffV + ftmpDiffV;
                            mDiff.insert(pair<int,double>(nOrgGear, fDiffV) );
                        }
                    }
                }
            }
        }
    }
    for(tmpGear = nGear; tmpGear >= nOrgGear ;tmpGear--){
        iRet = CheckNTimesHold( p1, p2, tmpGear, nOrgGear, fDiffV, 1 );
        if( iRet == -3 || iRet == 0 ){
            mDiff.insert(pair<int,double>(tmpGear, fDiffV) );
        }
    }
}

if( mDiff.empty() != true ){
    p_Diff = mDiff.begin();
    fDiffV = p_Diff->second;
    nSetGear = 0;
    // find small diff
    for( p_Diff = mDiff.begin(); p_Diff != mDiff.end();
                        p_Diff++ ){
        if( fabs( p_Diff->second ) <= fabs( fDiffV ) ){
            fDiffV = fabs( p_Diff->second);
        }
    }
    //find gear
    for( p_Diff = mDiff.begin(); p_Diff != mDiff.end();
                        p_Diff++ ){
        if( fabs(p_Diff->second) == fDiffV ){
            if( iMode == 0 ){
                if( nSetGear == 0 ){
                    nSetGear = p_Diff->first;
```

```
                }else{
                    if( nSetGear > p_Diff->first ){
                        nSetGear = p_Diff->first;
                    }
                }
            }else{
                if( nSetGear == 0 ){
                    nSetGear = p_Diff->first;
                }else{
                    if( nSetGear < p_Diff->first ){
                        nSetGear = p_Diff->first;
                    }
                }
            }
        }
    }
    mDiff.erase( mDiff.begin(), mDiff.end() );
    mDiff.clear();
    return true;
}else{
    return false;
}

}
/**/
/*****************************************************************
 * Function name           : CalcGear
 * Function summary        : Calculate gear setting module
 * Explanation             : The candidate of the gear at the gear holding time is decided.
 *                         :
 * Argument (input)        : p1         : Calculate start position
 * Argument (input)        : nOrgGrear  : Gear position (calculate origin)
 * Argument (output)       : nGrear     : Gear position (calculate object)
 * Argument (I/O)          : None
 * Return value            : true : Normal   false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool    CForwardCalc::CalcGear( map<double, stData>::iterator p1,
                                int &nGear, int nOrgGear, int iMode)
{
    bool bRet;
    int tmpGear;
    int tmpMaxGear;
    int nSetGear;
    int nOffsetGear;
    double  fNeMinLimit;                        // Engine speed determination lower limit
    double  fNeMaxTopLimit;                     // Max. engine speed rate
    stExF tmpExF;                               // Structure for search
    stData tmpData;                             // Temporary analysis data
    map<double,stData>::iterator p_tmpData;     // Temporary pointer
    map<double,stData>::iterator p_befData;     // Temporary pointer
    map<double,stData>::iterator p2;
    int iMiniGear;
    int iCnt;
    int iRet;

    // The gear before raising a shift is held.
    tmpGear = nOrgGear;

    memset( &tmpExF, 0x00, sizeof( tmpExF ));   // Initializes search structure.
    tmpExF.nGear = nGear;                       // Sets gear.
    p_ExF = m_ExF.find( tmpExF );               // Obtains applicable gear position.
    fNeMinLimit = p_ExF->fMinNe;                // Sets lower-limit engine speed.
    fNeMaxTopLimit = m_fOutRot;

    nOffsetGear = 3;
    if( m_fSubGearRatio.empty() != true ){
        nOffsetGear = nOffsetGear * m_fSubGearRatio.size();
    }
    if( tmpGear + nOffsetGear > m_nMaxGear ){
        tmpMaxGear = m_nMaxGear;
    }else{
        tmpMaxGear = tmpGear +nOffsetGear;
    }

    iMiniGear = m_nInitGear;

    p_befData = p1;
    if( p_befData != mp_PtnData[m_ModeID].begin() ){
        p_befData--;                                    // Preceding speed
    }

    if( p_befData->second.nGearTime >= 3 ){
        bRet = false;
        if( iMode == 0 ){
            for( tmpGear = tmpMaxGear; tmpGear >= nOrgGear; tmpGear-- ){
```

```
        bRet = CheckNowShiftChange( p1, tmpGear, nOrgGear, iMode );
        if( bRet == true ){
            bRet = Check3TimesHold( p1, tmpGear, nOrgGear, iMode );
            if( bRet == true ){
                nGear = tmpGear;
                return true;
            }
            bRet = false;
        }
    }

    for( tmpGear = tmpMaxGear; tmpGear >= nOrgGear; tmpGear-- ){
        bRet = CheckNowShiftChange( p1, tmpGear, nOrgGear, iMode );
        if( bRet == true ){
            p_tmpData = p1;
            iCnt = (int)GEAR_HOLD_TIME;
            nSetGear = 0;
            // Search for prospective gear for conditions.
            memset( &tmpData, 0x00, sizeof( tmpData )); // Initializes temporary data.
            tmpData.fTimes = p1->second.fTimes + iCnt ;     // Sets gear hold time for shift-up.

            p2 = mp_PtnData[m_ModeID].find( tmpData.fTimes );           // Up to end of gear hold time

            bRet = CheckNTimesSpeedHold( p1, p2, tmpGear, nOrgGear, nSetGear, 2);
            if( bRet == true ){
                nGear = nSetGear;
                return true;
            }
            if(( nSetGear != 0 )&&( nSetGear >= tmpGear )){
                nGear = nSetGear;
                return true;
            }
            bRet = false;
        }
    }

}else if( iMode == 1 ){
    for( tmpGear = nOrgGear; tmpGear >= iMiniGear; tmpGear-- ){
        bRet = CheckNowShiftChange( p1, tmpGear, nOrgGear, iMode );
        if( bRet == true ){
            bRet = Check3TimesHold( p1, tmpGear, nOrgGear, iMode );
            if( bRet == true ){
                break;
            }
        }
    }
}else if( iMode == 2 ){
    for( tmpGear = tmpMaxGear; tmpGear >= iMiniGear; tmpGear-- ){
        bRet = CheckNowShiftChange( p1, tmpGear, nOrgGear, iMode );
        if( bRet == true ){
            bRet = Check3TimesHold( p1, tmpGear, nOrgGear, iMode );
            if( bRet == true ){
                break;
            }
        }
    }
}else if( iMode == 3 ){
    for( tmpGear = tmpMaxGear; tmpGear >= iMiniGear; tmpGear-- ){
        bRet = CheckNowShiftChange( p1, tmpGear, nOrgGear, iMode );
        if( bRet == true ){
            p_tmpData = p1;
            iCnt = (int)GEAR_HOLD_TIME;
            nSetGear = 0;
            // Search for prospective gear for conditions.
            memset( &tmpData, 0x00, sizeof( tmpData )); // Initializes temporary data.
            tmpData.fTimes = p1->second.fTimes + iCnt ;     // Sets gear hold time for shift-up.

            p2 = mp_PtnData[m_ModeID].find( tmpData.fTimes );           // Up to end of gear hold time

            bRet = CheckNTimesSpeedHold( p1, p2, tmpGear, iMiniGear, nSetGear, 2);
            if( bRet == true ){
                nGear = nSetGear;
                return true;
            }
            if( nSetGear != 0 ){
                nGear = nSetGear;
                return true;
            }
            bRet = false;
        }
    }
}

if( bRet == true ){
    nGear = tmpGear;
    if( nGear != nOrgGear ){
        return true;
    }
```

```
    }
}

nGear = nOrgGear;
// Calculate acceleration, torque, and engine speed for this time.
bRet = pCalPos->CalcNeGear( p1, nGear );        // Engine speed
if( bRet != true ){
    cout << "(" << __LINE__ << ") ";
    return false;
}

bRet = pCalPos->CalcVanaGear( p1, nGear );  // An analyzed car speed is calculated.
if( bRet == false ){
    cout << "(" << __LINE__ << ") ";
    return false;
}

bRet = pCalPos->CalcTqGear( p1, nGear );        // Engine torque (without complement)
if( bRet == false ){
    cout << "(" << __LINE__ << ") ";
    return false;
}

p1->second.fMaxTe = pCalPos->GetLRevMaxTq( p1->second.fNegrevo );
p1->second.fCarA =  ( p1->second.fVref_sp - p_befData->second.fVana_sp ) *
                        1000.0/(60.0*60.0);

if( p1->second.fNegrevo >= fNeMaxTopLimit ){
    p_tmpData = p1;
    for( iCnt = (int)GEAR_HOLD_TIME; iCnt >= 1; iCnt--){    // All sections subject to determination
        // Search for prospective gear for conditions.
        memset( &tmpData, 0x00, sizeof( tmpData )); // Initializes temporary data.
        tmpData.fTimes = p1->second.fTimes + iCnt ;     // Sets gear hold time for shift-up.

        p2 = mp_PtnData[m_ModeID].find( tmpData.fTimes );           // Up to end of gear hold time

        bRet = CheckNTimesSpeedHold( p_tmpData, p2, nGear, nOrgGear, nSetGear, 0);
        if( bRet == true ){
            nGear = nSetGear;
            break;
        }
    }
    return true;
}

if( p1->second.fNegrevo <= fNeMinLimit){        // In case of min. engine speed or less
    p_tmpData = p1;
    bRet = pCalPos->CalcMaxSp( p_tmpData,
                               nGear );           // Target-speed follow calculation processing
    if( bRet == false ){
        cout << "(" << __LINE__ << ") ";
        return -99;
    }
    for( iCnt = (int)GEAR_HOLD_TIME; iCnt >= 1; iCnt--){    // All sections subject to determination
        // Search for prospective gear for conditions.
        memset( &tmpData, 0x00, sizeof( tmpData )); // Initializes temporary data.
        tmpData.fTimes = p1->second.fTimes + iCnt;      // Sets gear hold time for shift-up.

        p2 = mp_PtnData[m_ModeID].find( tmpData.fTimes );           // Up to end of gear hold time

        bRet = CheckNTimesSpeedHold( p_tmpData, p2, nGear, nOrgGear, nSetGear, 1);
        if( bRet == true ){
            nGear = nSetGear;
            break;
        }
    }

    if( bRet == false ){
        nGear = m_nDownLimitGear;
    }
    return true;
}

if( p1->second.fMaxTe < p1->second.fTe ){
    p_tmpData = p1;
    bRet = pCalPos->CalcMaxSp( p_tmpData,
                               nGear );           // Target-speed follow calculation processing
    if( bRet == false ){
        cout << "(" << __LINE__ << ") ";
        return -99;
    }
    for( iCnt = (int)GEAR_HOLD_TIME; iCnt >= 1; iCnt--){    // All sections subject to determination
        // Search for prospective gear for conditions.
        memset( &tmpData, 0x00, sizeof( tmpData )); // Initializes temporary data.
        tmpData.fTimes = p1->second.fTimes + iCnt;      // Sets gear hold time for shift-up.

        p2 = mp_PtnData[m_ModeID].find( tmpData.fTimes );           // Up to end of gear hold time
```

```
            bRet = CheckNTimesSpeedHold( p_tmpData, p2, nGear, nOrgGear, nSetGear, 3);
            if( bRet == true ){
                nGear = nSetGear;
                break;
            }
        }
    }



    return true;
}

/**/
/*****************************************************************
 * Function name           : CalcStFollow
 * Function summary        : Calculate startup gear setting module
 * Explanation             : A gear in starting is set up.
 *                         :
 * Argument (input)        : None
 * Argument (input)        : nGear : Gear position
 * Argument (output)       : None
 * Argument (I/O)          : p1 : Calculate position
 * Return value            : true : Normal   false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool    CForwardCalc::CalcStFollow( map<double, stData>::iterator &p1 )
{
    map<double,stData>::iterator p_start;       // Temporary pointer
    map<double,stData>::iterator p_bef;         // Temporary pointer
    double  fNegrevo;                           // Engine speed
    double  fTe;                                // Engine torque
    double  fTe_def;                            // Engine torque
    int     tmpGear;
    bool    bRet;
    double  fCarV;
    double  fGearRaito;
    bool    bStartIsAlready;
    int     nSetGear;
    double  fDiffV;
    map<int,double>           mpDiff;
    map<int,double>::iterator p_mpDiff;

    bStartIsAlready = false;

    p_start = p1;

    if(( p1 == mp_PtnData[m_ModeID].begin() )&&( p1->second.fVref_sp != 0 )){
        p1->second.fVana_sp = p1->second.fVref_sp;
        bStartIsAlready = true;
        p1->second.nGearTime = 4;
        tmpGear = m_nMaxGear;
        bRet = CalcGear( p1, tmpGear, m_nMaxGear, 2 );
    }else{
        p1->second.nGear = m_nInitGear;
        tmpGear = m_nInitGear;
    }

CHECK_AGAIN:

    // Calculate vehicle speed.
    bRet = pCalPos->GetGearFullRatio( tmpGear, fGearRaito);
    if( bRet != true ){
        return false;
    }
    fCarV = (2.0 * PI * m_fTarR )/ 1000.0 * 60.0 * m_fCl_MetNe / fGearRaito;            // Time for clutch to engage


    for( ; ; p1++ ){
        // Calculate acceleration, torque, and engine speed for this time.
        bRet = pCalPos->CalcNeGear( p1, tmpGear );  // Engine speed
        if( bRet != true ){
            cout << "(" << __LINE__ << ") ";
            return false;
        }
        fNegrevo = p1->second.fNegrevo;

        bRet = pCalPos->CalcVanaGear( p1, tmpGear );// An analyzed car speed is calculated.
        if( bRet == false ){
            cout << "(" << __LINE__ << ") ";
            return false;
        }

        if( fNegrevo < m_fCl_MetNe ){
            fNegrevo = m_fCl_MetNe;
        }
```

```
    bRet = pCalPos->CalcTqGear( p1, tmpGear );  // Engine torque (without complement)
    if( bRet == false ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }

    p1->second.fMaxTe = pCalPos->GetLRevMaxTq( fNegrevo );
    if( p1->second.fTe > p1->second.fMaxTe ){   // Current status maintenance, shift-down
      if( bStartIsAlready != true ){
        if( tmpGear >= 1 ){
            tmpGear--;
            if( tmpGear < 1 ){
                tmpGear = 1;
                break;
            }
            p1 = p_start;
            goto CHECK_AGAIN;
        }
      }
    }

    p1->second.fNegrevo = fNegrevo;

    bRet = pCalPos->CalcMaxSp( p1, tmpGear,1 ); // Target-speed follow calculation processing
    if( bRet == false ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }

    p1->second.nGearTime = 4;
    p1->second.nGear = tmpGear;

    p_bef = p1;
    if( p_bef != mp_PtnData[m_ModeID].begin() ){
        p_bef--;
    }

    if( p_bef->second.fVana_sp > fCarV ){
        mpDiff.insert(pair<int,double>(tmpGear,fabs( p1->second.fVana_sp - p1->second.fVref_sp) ) );

        if( tmpGear >= 1 ){
            tmpGear--;
            if( tmpGear < 1 ){
                break;
            }
            p1 = p_start;
            goto CHECK_AGAIN;
        }
    }
    if( p1 == mp_PtnData[m_ModeID].end() ){
        mpDiff.insert(pair<int,double>(tmpGear,fabs( p1->second.fVana_sp - p1->second.fVref_sp) ) );

        if( tmpGear >= 1 ){
            tmpGear--;
            if( tmpGear < 1 ){
                break;
            }
            p1 = p_start;
            goto CHECK_AGAIN;
        }
    }

    // ----------------------------------------
    // If 5% normalized engine speed reached
    // ----------------------------------------
    if( fNegrevo > m_fCl_MetNe ){
        mpDiff.insert(pair<int,double>(tmpGear,fabs( p1->second.fVana_sp - p1->second.fVref_sp) ) );

        if( tmpGear >= 1 ){
            tmpGear--;
            if( tmpGear < 1 ){
                break;
            }
            p1 = p_start;
            goto CHECK_AGAIN;
        }
    }

    bRet = pCalPos->CalcAllData( p1 );          // Calculate All Data
    if( bRet == false ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }

}

// set gear
```

```
if( mpDiff.empty() != true ){
    p_mpDiff = mpDiff.begin();
    nSetGear = p_mpDiff->first;
    fDiffV = p_mpDiff->second;
    for( p_mpDiff = mpDiff.begin(); p_mpDiff != mpDiff.end(); p_mpDiff++ ){
        if( fDiffV > p_mpDiff->second ){
            nSetGear = p_mpDiff->first;
            fDiffV = p_mpDiff->second;
        }else if( fDiffV == p_mpDiff->second ){
            if( nSetGear < p_mpDiff->first ){
                nSetGear = p_mpDiff->first;
                fDiffV = p_mpDiff->second;
            }
        }
    }

    tmpGear = nSetGear;
}else{
    tmpGear = 1;
}

// Calculate vehicle speed.
bRet = pCalPos->GetGearFullRatio( tmpGear, fGearRaito);
if( bRet != true ){
    return false;
}
fCarV = (2.0 * PI * m_fTarR )/ 1000.0 * 60.0 * m_fCl_MetNe / fGearRaito;            // Time for clutch to engage

p1 = p_start;
for( ; ; p1++ ){
    bRet = pCalPos->CalcNeGear( p1, tmpGear );  // Engine speed
    if( bRet != true ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }
    fNegrevo = p1->second.fNegrevo;

    bRet = pCalPos->CalcVanaGear( p1, tmpGear );// An analyzed car speed is calculated.
    if( bRet == false ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }

    if( fNegrevo < m_fCl_MetNe ){
        fNegrevo = m_fCl_MetNe;
    }

    bRet = pCalPos->CalcTqGear( p1, tmpGear );  // Engine torque (without complement)
    if( bRet == false ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }

    p1->second.fMaxTe = pCalPos->GetLRevMaxTq( fNegrevo );
    if( bRet == false ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }

    p1->second.fNegrevo = fNegrevo;

    bRet = pCalPos->CalcMaxSp( p1, tmpGear,1 ); // Target-speed follow calculation processing
    if( bRet == false ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }

    p1->second.nGearTime = 4;
    p1->second.nGear = tmpGear;

    bRet = pCalPos->CalcAllData( p1 );          // Calculate All Data
    if( bRet == false ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }

    p_bef = p1;
    if( p_bef != mp_PtnData[m_ModeID].begin() ){
        p_bef--;
    }

    if( p_bef->second.fVana_sp > fCarV ){
        break;
    }
    if( p1 == mp_PtnData[m_ModeID].end() ){
        break;
    }
```

```
        // ---------------------------------------
        // If 5% normalized engine speed reached
        // ---------------------------------------
        if( fNegrevo > m_fCl_MetNe ){
            break;
        }

    }
    return true;
}
/**/
/*****************************************************************
 * Function name           : CExecuteProc
 * Function summary        : Constructor
 * Explanation             : Class constructor
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
CExecuteProc::CExecuteProc()
{
    pReadF  = new CFileIO;
    pCalPos = new CCalculatePos;
    pFCal   = new CForwardCalc;

    pFileIO = new CFileIO;

    pConsole= new CConsoleIO;

    return;
}

/**/
/*****************************************************************
 * Function name           : ~CExecuteProc
 * Function summary        : Destructor
 * Explanation             : Class destructor
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
CExecuteProc::~CExecuteProc()
{
    delete pReadF;
    delete pCalPos;
    delete pFCal;

    delete pFileIO;
    delete pConsole;

    return;
}

/**/
/*****************************************************************
 * Function name           : Init
 * Function summary        : Initialize
 * Explanation             : The initialization management of the executive process
 *                         :
 * Argument (input)        : None
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool CExecuteProc::Init(int argc, char* argv[])
{
    bool bRet;

    // ----------------------
    // Retreive data from arguments parameters
    // ----------------------
    bRet = Init_InputArguments(argc, argv);
    if(bRet != true ){
        cout << "(" << __LINE__ << ") ";
        return false;
```

```
    }

    // ----------------------
    // Retreive data from input files
    // ----------------------
    bRet = Init_InputData();
    if(bRet != true ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }

    // ----------------------
    // Initialize values that will be used during calculation proces
    // ----------------------
    bRet = Init_Caracteristics();
    if(bRet != true ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }


    bRet = pCalPos->GetExF();
    if(bRet != true ){
        cout << "(" << __LINE__ << ") ";
        return false;
    }

    if(m_vd1rev.empty()!=true){
        m_vd1rev.erase( m_vd1rev.begin(), m_vd1rev.end() );
        m_vd1rev.clear();
    }
    if(m_vd1trq.empty() != true ){
        m_vd1trq.erase( m_vd1trq.begin(), m_vd1trq.end() );
        m_vd1trq.clear();
    }
    // 20X20 mapcreate
    fuelInit( (char *)m_FuelConsumFile.c_str(), (char *)m_MaxLossFile.c_str(), 20, 20 );

    return true;
}

/**/
/*****************************************************************
 * Function name           : InitValues
 * Function summary        : Environmental data read module
 * Explanation             : Environmental file is read and set in parameters.
 *                         :
 * Argument (input)        : None
 * Argument (output)               : None
 * Argument (I/O)          : None
 * Return value            : true : Normal  false : Failure
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool CExecuteProc::Init_Caracteristics()
{
    stData tmpData;

    map<double,stData> mpData;// Analysis data table

    // ************************************************//
    // Preparation of MODE_1 datas                     //
    // ************************************************//
    // Reset of mpDatas
    if( !mpData.empty() ){
        mpData.erase( mp_PtnData[m_ModeID].begin(), mp_PtnData[m_ModeID].end() );
        mpData.clear();
    }
    // Conversion of MODE_1 datas to vector
    for(int i=0; i<MODE1_SIZE;i++){
        //Reset of "tmpData" structure
        memset(&tmpData, 0x00, sizeof(tmpData));

        tmpData.fTimes  = mode1[i][0]+1; //Time
        tmpData.fV      = mode1[i][1]; //Speed
        tmpData.fGrade  = mode1[i][2]; //Grade
        tmpData.fVref_sp= tmpData.fV;
        // Stores environment datas
        mpData.insert(pair<double,stData>(tmpData.fTimes, tmpData));
    }
    mp_PtnData.insert( pair<int, map<double,stData> >(1, mpData ) );
    m_vModeNames.push_back(nameMode1);


    // ************************************************//
    // Preparation of MODE_2 datas                     //
    // ************************************************//
```

```
// Reset of mpDatas
if( !mpData.empty() ){
    mpData.erase( mp_PtnData[m_ModeID].begin(), mp_PtnData[m_ModeID].end() );
    mpData.clear();
}
//------------------------------
// Conversion of MODE_2 datas to vector
//------------------------------
for(int i=0; i<MODE2_SIZE;i++){
    //Reset of "tmpData" structure
    memset(&tmpData, 0x00, sizeof(tmpData));

    tmpData.fTimes  = mode2[i][0]+1; //Time
    tmpData.fV      = mode2[i][1]; //Speed
    tmpData.fGrade  = mode2[i][2]; //Grade
    tmpData.fVref_sp= tmpData.fV;
    // Stores environment datas
    mpData.insert(pair<double,stData>(tmpData.fTimes, tmpData));
}
mp_PtnData.insert( pair<int, map<double,stData> >(2, mpData ) );
m_vModeNames.push_back(nameMode2);



// ***********************************************//
// Initialisation of calculus datas               //
// ***********************************************//
m_fCarCurbW   = divValues[m_iConsDivisionIndex][0]*1000;        // Curb vehicle weight (empty vehicle mass (kg))

if(divValues[m_iConsDivisionIndex][1]==NULL)
{
    m_nCarKind=BUS;
    m_fCarMaxPayload = m_fCarCurbW+ (m_fPersons+1)*PERSON_W;
}
else{
    m_nCarKind=TRACTOR;
    m_fCarMaxPayload = divValues[m_iConsDivisionIndex][1]*1000;        // Max load of vehicle
}

m_fPersons   = divValues[m_iConsDivisionIndex][2];        // Riding capacity information
m_fOverHeight= divValues[m_iConsDivisionIndex][3];        // Overall vehicle height
m_fOverWidth = divValues[m_iConsDivisionIndex][4];        // Overall vehicle width

m_fGrossVehicleWeight = m_fCarMaxPayload + m_fCarCurbW + (PERSON_W * m_fPersons);


// Starting gear
m_nInitGear=(int)m_vTransmissionData[0];

// ----------------------
// Reads gear number and ratio for main transmission gear
// ----------------------
m_nMaxMainGear=(int)m_vTransmissionData[1];        // Number of gears for main transmission gear
for( int i = 1; i <= m_nMaxMainGear; i++ ){
    m_fMainGearRatio.push_back((double)m_vTransmissionData[1+i]); // Stores gear ratio.
}

// ----------------------
// Reads gear number and ratio for second transmission gear
// ----------------------
m_nMaxSubGear= (int)m_vTransmissionData[2+m_nMaxMainGear];
for( int i = 1; i <= m_nMaxSubGear; i++ ){
    m_fSubGearRatio.push_back((double)m_vTransmissionData[2+i+m_nMaxMainGear]); // Stores gear ratio.
}

m_nTorqueConv = (int)m_vTransmissionData[(int)m_vTransmissionData.size()-1];

//---------------------------
// In case of no "subGear", m_fMainGearRatio datas are copied into m_fGearRatio
//---------------------------
if( m_fSubGearRatio.empty() ){  // Copy of m_fMainGearRatio content into  m_fGearRatio
    for( int i = 1; i <= (int)m_fMainGearRatio.size(); i++ ){
        m_fGearRatio.push_back( m_fMainGearRatio[i-1] );
    }

    m_nDownLimitGear = m_nInitGear;
    m_nMaxGear = m_fMainGearRatio.size();
}
else{
//---------------------------
// When there is a "subGear", m_fGearRatio get m_fMainGearRatio * m_fSubGearRatio
//---------------------------
    for( int i = 1; i <= (int)m_fMainGearRatio.size(); i++ ){
        for( int j = 1; j <= (int)m_fSubGearRatio.size(); j++ ){
            m_fGearRatio.push_back( m_fMainGearRatio[i-1] * m_fSubGearRatio[j-1] );
        }
    }
```

```
        m_nDownLimitGear = m_nInitGear * m_fSubGearRatio.size();
        m_nInitGear = m_nInitGear * m_fSubGearRatio.size();
        m_nMaxGear = m_fMainGearRatio.size() * m_fSubGearRatio.size();
    }

    // ***********************************************//
    // Setting of Vehicle weight                      //
    // ***********************************************//
    if(m_nCarKind==1)// BUS
        m_fCarWeight = m_fCarCurbW + m_fPersons * PERSON_W / 2.0;

    else if(m_nCarKind==2)// TRACTOR
        m_fCarWeight = m_fCarCurbW + m_fCarMaxPayload/2 + PERSON_W;

    else //TRUCK weight is the default value
        m_fCarWeight = m_fCarCurbW + m_fCarMaxPayload/2 + PERSON_W;


    // ***********************************************//
    // Setting of Rolling resistance and aerodynamic drag coefficients
    // ***********************************************//
    m_fMuAir = (0.00299*m_fOverWidth*m_fOverHeight-0.000832);
    m_fMuRollRes = 0.00513 + (17.6/m_fCarWeight);

    if(m_nCarKind==BUS) m_fMuAir *= 0.680; //μaA=(0.00299B･H-0.000832)×K


    // ***********************************************//
    // Setting of clutch meet and release revolutions  //
    // ***********************************************//
    m_fCl_RelNe= CLUTCH_RELEASE * (m_fFixedNe-m_fIdleNe)/100.0+m_fIdleNe;  // Calculates clutch release engine speed.
    m_fCl_MetNe= CLUTCH_MEET * (m_fFixedNe-m_fIdleNe)/100.0+m_fIdleNe;     // Calculates clutch meet engine speed.

    return true;
}

/**/
/*****************************************************************
 * Function name           : CalculateStart
 * Function summary        : Calculate start
 * Explanation             : The executive management of the executive process
 *                         :
 * Argument (input)        : Pattern mode file index
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : None
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
bool CExecuteProc::CalculateStart(int Index)
{
    bool bRet;
    int  i;
    int nbShiGaiChiValues=0;
    double tmpDist, tmpShigaichiDist;
    double tmp_SumFuel, tmp_SumShiGaiChiFuel;

    map<double, stData>::iterator p_bef;

    if( Index<0 || Index>3 || Index>(int)mp_PtnData.size()){
        cout << "Index value is not correct (" << Index << ")";
        return false;
    }

    // Average&Sum Clear
    m_AveSpeed[Index-1]=0.0;       // Average Speed
    m_AveFuel[Index-1]=0.0;        // Average Fuel

    tmp_SumFuel  = 0.0;
    tmp_SumShiGaiChiFuel = 0.0;

    tmpDist     = 0.0;
    tmpShigaichiDist = 0.0;

    // Calculation management start
    for(p_mpData=mp_PtnData[Index].begin(); p_mpData != mp_PtnData[Index].end(); p_mpData++ ){


        bRet = pFCal->CalcGearCheck( p_mpData );            // Calculate gear setting main module
        if(bRet != true ){
            cout << "Error.(GearCheck)";
            return false;
        }

        bRet = pCalPos->CalcAllData(p_mpData);          // Calculate All Data
        if(bRet != true ){
            cout << "Error.(AllData)";
```

```
            return false;
        }
    }

    for(p_mpData=mp_PtnData[Index].begin(); p_mpData != mp_PtnData[Index].end(); p_mpData++)
    {
        if(Index == 1){
            if((p_mpData->second.fTimes>=645) && (p_mpData->second.fTimes<1411)){
                nbShiGaiChiValues++;
                m_AveSpeed[2] += p_mpData->second.fVana_sp;
                tmp_SumShiGaiChiFuel += p_mpData->second.fFuel;
                tmpShigaichiDist += (p_mpData->second.fVana_sp);
            }
        }

        m_AveSpeed[Index-1] += p_mpData->second.fVana_sp;
        tmp_SumFuel+= p_mpData->second.fFuel;
        tmpDist += (p_mpData->second.fVana_sp);
    }


    //-------------------------
    // Calcul of average speed
    //-------------------------
    if(!mp_PtnData[Index].empty() && m_AveSpeed[Index-1] != 0.0){
        m_AveSpeed[Index-1] = m_AveSpeed[Index-1] / mp_PtnData[Index].size();
    }
    //-------------------------
    // Calcul of average fuel consumption
    //-------------------------
    m_AveFuel[Index-1] = 0.0;
    if(tmp_SumFuel != 0 && tmpDist != 0.0 ){
        m_AveFuel[Index-1] = tmpDist/tmp_SumFuel;
    }

    //-------------------------
    // Rectification of average fuel consumption in case of torque conversion asked
    //-------------------------
    if( m_nTorqueConv==1 ){// Torque convertion if needed
        if(Index==1) m_AveFuel[0] *= 0.91;
        if(Index==2) m_AveFuel[1] *= 0.96;
    }

    if(Index!=1) return true;


    //-------------------------
    // Calcul of average speed (shigaichi mode)
    //-------------------------
    if(!mp_PtnData[Index].empty() && m_AveSpeed[2] != 0.0){
        m_AveSpeed[2] = m_AveSpeed[2] / nbShiGaiChiValues;
    }
    //-------------------------
    // Calcul of average fuel consumption (shigaichi mode)
    //-------------------------
    m_AveFuel[2] = 0.0;
    if(tmp_SumShiGaiChiFuel != 0 && tmpDist != 0.0 ){
        m_AveFuel[2] = tmpShigaichiDist/tmp_SumShiGaiChiFuel;
    }
    if( m_nTorqueConv==1 ){// Torque convertion if needed
        m_AveFuel[2] *= 0.91;
    }
    return true;
}

/**/
/*****************************************************************
 * Function name           : main
 * Function summary        : Main processing
 * Explanation             : Main process of conversion processing
 *                         :
 * Argument (input)        : argc : input argument count
 * Argument (input)        : argv : input argument data
 * Argument (output)       : None
 * Argument (I/O)          : None
 * Return value            : 0 : Normal end
 * Created by              :
 * Updated on (created on) :
 * Remarks                 :
 *****************************************************************/
int main(int argc, char* argv[])
{
    bool bRet;
    char tmp[256];

    //----------------------
    // Initialization of calcul process
    //----------------------
```

```
pExecuteProc = new CExecuteProc;

//-----------------------
// Initialization of calculation environment
//-----------------------
bRet = pExecuteProc->Init(argc, argv);
if( bRet != true ){
    cout << "Error.(Init)";
    return false;
}

//-----------------------
// Display of vehicle caracteristics
//-----------------------
pExecuteProc->pConsole->DisplayCaracteristics();
if(!m_bSilentModeON)
    getch();

//-----------------------
// Display of results screen
//-----------------------
pExecuteProc->pConsole->DisplayResultsScreen();

//-----------------------
// Calculation process starts
//-----------------------
for(m_ModeID = 1; m_ModeID <= NB_RIDING_MODE; m_ModeID++ ){
    pExecuteProc->pConsole->myWriteConsole(10+(m_ModeID-1)*25, 2, "Calculating...");

    // Initiates conversion processing.
    bRet = pExecuteProc->CalculateStart( m_ModeID );
    if(bRet == false ){
        cout << "Stopped with error.(calculate)" << endl;
        exit(-1);
    }
    pExecuteProc->pConsole->myWriteConsole(10+(m_ModeID-1)*25, 2, "              ");


    // Display of Mode Name
    sprintf( tmp, "%s", m_vModeNames[m_ModeID-1].c_str());
    pExecuteProc->pConsole->myWriteConsole(10+(m_ModeID-1)*25, 2, tmp);

    // Display of average speed
    sprintf( tmp, "%5.1f(Km/h)", m_AveSpeed[m_ModeID-1]);
    pExecuteProc->pConsole->myWriteConsole(10+(m_ModeID-1)*25, 4, tmp);

    // Display of average fuel consumption
    sprintf( tmp, "%7.4f(km/l)", m_AveFuel[m_ModeID-1]);
    pExecuteProc->pConsole->myWriteConsole(8+(m_ModeID-1)*25, 5, tmp);

    if(m_ModeID==1){
        // Display of average speed
        sprintf( tmp, "%5.1f(Km/h)", m_AveSpeed[2]);
        pExecuteProc->pConsole->myWriteConsole(10+2*25, 4, tmp);

        // Display of average fuel consumption
        sprintf( tmp, "%7.4f(km/l)", m_AveFuel[2]);
        pExecuteProc->pConsole->myWriteConsole(8+2*25, 5, tmp);
    }
}

//-----------------------
// Calcul average fuel consumption for the two modes)
//-----------------------
double alpha=divValues[m_iConsDivisionIndex][5]/100;
double Fu=m_AveFuel[0];
double Fh=m_AveFuel[1];

m_fFuelAverageCons=1/((1-alpha)/Fu + alpha/Fh);

//-----------------------
// Display of average fuel consumption
//-----------------------
sprintf( tmp, "%.4f km/l", m_fFuelAverageCons);
pExecuteProc->pConsole->myWriteConsole(30, 8, tmp);

//-----------------------
// Display of output file name
//-----------------------
if(m_OutputData=="")
    pExecuteProc->pConsole->ReadConsole(17, 10, 30, m_OutputData, ALLOWED_CAR);

//-----------------------
// Output results
//-----------------------
bRet = pExecuteProc->pFileIO->WriteCalculateData();
if(bRet == false ){
    cout << "Stopped with error.(File Output)" << endl;
```

```
        exit(-1);
    }

    //------------------------
    // End of process
    //------------------------
    pExecuteProc->pConsole->myWriteConsole(3, 18, "                                          ");
    pExecuteProc->pConsole->myWriteConsole(3, 18, "Process succefully completed.");

    if(!m_bSilentModeON)
        getch();

    // Post-processing
    delete pExecuteProc;

    exit(0);
    return(0);
}
//---------------------------------------------------------------------------
#endif
```

```
#include "CCsvFile.h"

//-------------------------------------------------
//  CCsvFile.cpp
//
//  概要： CSVファイル形式のデータを読み込み、
//         データを保存、参照する。
//
//-------------------------------------------------


//-----------------------------------------
// コンストラクタ
//-----------------------------------------
CCsvFile::CCsvFile()
{
    return;
}

//-----------------------------------------
// デストラクタ
//-----------------------------------------
CCsvFile::~CCsvFile()
{
    //----------------------------
    // メモリ内容をクリアする。
    //----------------------------
    DataClear();

    return;
}

//-----------------------------------------
// メモリ内容クリア処理
//-----------------------------------------
void CCsvFile::DataClear(void)
{
    //-------------------------------------------------
    // 読み込みファイルのメモリ領域削除処理
    //-------------------------------------------------
    if(!vCsvLineData.empty()){
        for(p_vCsvLineData=vCsvLineData.begin();p_vCsvLineData!=vCsvLineData.end();p_vCsvLineData++)
        {// ファイル読み込み行数分ループ
            if(!p_vCsvLineData->word.empty()){// １行データがある場合
                p_vCsvLineData->word.erase( p_vCsvLineData->word.begin(),p_vCsvLineData->word.end());
                p_vCsvLineData->word.clear();// １行データのクリア
            }
            if(!p_vCsvLineData->type.empty()){// １行データがある場合
                p_vCsvLineData->type.erase( p_vCsvLineData->type.begin(),p_vCsvLineData->type.end());
                p_vCsvLineData->type.clear();// １行データのクリア
            }
        }

        vCsvLineData.erase(vCsvLineData.begin(),vCsvLineData.end());//１行の完全削除
        vCsvLineData.clear();                                       // コンテナの削除
    }

    return;
}
//-----------------------------------------
// File reading
//-----------------------------------------
bool CCsvFile::ReadFile(string strFileName )
{
    string tmpStr;     // １行読み込み文字列
    FILE *fp;
    char *p, *pchk;
    char wkstr[1024];
    char chkstr[1024];

    // ファイルを読み込みオープンする。
    fp = fopen( strFileName.c_str(), "r" );

    if((fp == NULL)||(ferror(fp))) return false; // ファイルオープン失敗

    //----------------------------
    // 内部データのクリア
    //----------------------------
    DataClear();

    //----------------------------
    // Reading each line of "DATA" file and storing its datas into vCsvLineData object
    //----------------------------
    while(!feof(fp)){
        memset( wkstr, 0x00, sizeof( wkstr ));
        p = fgets( wkstr, 1023, fp ); // １行読み込み
        if( p == NULL ) break;
        if( ferror(fp) ) break;
```

```
        // １行データの設定
        tmpStr = string( wkstr );
        //append 2007.01.23 tmpStr check
        memset( chkstr, 0x00, sizeof(chkstr));
        memcpy( chkstr, wkstr, strlen(wkstr)-1 );
        pchk = strtok( chkstr, " \n" );
        if(( tmpStr != "" )&&( pchk != NULL )){
            SetDataStr(tmpStr);
        }
    }

    fclose(fp);

    return true;
}
//---------------------------------------
// ファイル書き込み
//---------------------------------------
bool CCsvFile::WriteFile(string filename, char delm)
{
    int tmp_sizepos1,tmp_sizepos2,tmp_sizepos3;
    string tmpStr;                                          // １行読み込み文字列
    vector<string>::iterator p_str;
    vector<int>::iterator p_int;
    FILE *fp;

    // ファイルを書き込みオープンする。
    fp = fopen( filename.c_str(), "w" );

    if((fp == NULL)||(ferror(fp))) return false;// ファイルオープン失敗

    for(p_vCsvLineData=vCsvLineData.begin();p_vCsvLineData!=vCsvLineData.end();p_vCsvLineData++)
    {// 行数分ループ
        for(p_str=p_vCsvLineData->word.begin(),p_int=p_vCsvLineData->type.begin();p_str!=p_vCsvLineData->word.end();)
        {// カラム数分ループ
            tmpStr = *p_str;
            tmp_sizepos1 = (int)(tmpStr.find_first_of( ' ', 0 ));
            tmp_sizepos2 = (int)(tmpStr.find_first_of( '\t', 0 ));
            tmp_sizepos3 = (int)(tmpStr.find_first_of( ',', 0 ));

            if((tmp_sizepos1>=0)||(tmp_sizepos2>=0)||(tmp_sizepos3>=0)||(*p_int==1))
            {
                fprintf(fp, "\"%s\"", p_str->c_str() );// データを書き込みする。
            }
            else{
                fprintf( fp, "%s", p_str->c_str() );// データを書き込みする。
            }

            p_str++;                                        // 次のポインタに移動
            p_int++;
            if( p_str != p_vCsvLineData->word.end() ){// 最終カラムでない場合
                fprintf( fp, "%c", delm );// 区切り文字を入れる
            }
        }

        fprintf( fp, "\n" );

    }

    fclose(fp);

    return true;
}
//---------------------------------------
// レコード数の取得
//---------------------------------------
int  CCsvFile::RecSize(void)
{
    return( (int)(vCsvLineData.size()) );
}
//---------------------------------------
// Locate the 'p_vCsvLineData' pointer at the 'index' position
//---------------------------------------
bool CCsvFile::SetAtRec(int index )
{
    int i;

    if(((int)(vCsvLineData.size())>=index) && (index>0)){
        for( p_vCsvLineData = vCsvLineData.begin(), i = 1;i != index; i++ ){
            p_vCsvLineData++;
        }
    }
    else{
        p_vCsvLineData = vCsvLineData.end();
        return false;
    }
```

```
    return true;
}
//-------------------------------------
// １行データの設定
//-------------------------------------
bool CCsvFile::SetDataStr(string str)
{
    stCsvLineData tmpCsvLineData;
    char wkstr[256];                // 読み込み文字列バッファ
    char wkstr_wd[256];
    int i;
    int tmp_delm;                   // 区切り区間でないかのフラグ
    int tmp_delmIndex;              // 区切り位置の開始位置
    string tmpStr;

    if(!tmpCsvLineData.word.empty()){
        tmpCsvLineData.word.erase(tmpCsvLineData.word.begin(),tmpCsvLineData.word.end());
        tmpCsvLineData.word.clear();

        tmpCsvLineData.type.erase(tmpCsvLineData.type.begin(),tmpCsvLineData.type.end());
        tmpCsvLineData.type.clear();
    }

    // １文字づつ取得し、チェックする。
    tmp_delm = 0;
    tmp_delmIndex = 0;

    sprintf( wkstr, "%s", str.c_str() );
    for( i = 0; wkstr[i] != 0x00; i++ ){
        if( wkstr[i] == '"' ){
            // 区切り区間開始？
            if( tmp_delm == 0 ){    // 区切り区間開始発生
                tmp_delm = 1;       // 区切り位置を設定
                i++;
                tmp_delmIndex=i;    // 区切り開始位置を設定
            }
            else{                                               // 区切り区間終了？
                memset(wkstr_wd, 0x00, sizeof(wkstr_wd));
                memcpy(wkstr_wd, &wkstr[tmp_delmIndex],i-tmp_delmIndex); // 区切り位置を設定する。
                tmpStr = string( wkstr_wd );
                tmpCsvLineData.word.push_back( tmpStr );
                tmpCsvLineData.type.push_back( tmp_delm );
                tmp_delm = 0;
                // 次の開始位置を設定する。
                i++;
                tmp_delmIndex = i;
                while(1){
                    if( wkstr[i] == 0x00 ){
                        i--;
                        break;
                    }

                    if((wkstr[i]==' ') ||(wkstr[i]=='\t')||
                       (wkstr[i]=='\n')||(wkstr[i]==',')){
                        i++;
                        tmp_delmIndex = i;
                    }
                    else{
                        i--;
                        break;
                    }
                }
            }
        }
        else if((wkstr[i]==' ') ||(wkstr[i]=='\t')||
                (wkstr[i]=='\n')||(wkstr[i]==',')){

            // 通常の区切り位置発見？
            if(tmp_delm!=1){
                memset(wkstr_wd, 0x00, sizeof(wkstr_wd));
                memcpy(wkstr_wd, &wkstr[tmp_delmIndex],i-tmp_delmIndex); // 区切り位置を設定する。
                tmpStr = string( wkstr_wd );

                tmpCsvLineData.word.push_back( tmpStr );
                tmpCsvLineData.type.push_back( tmp_delm );

                // 次の開始位置を設定する。
                tmp_delmIndex = i;
                while(1){
                    if(wkstr[i]==0x00){
                        i--;
                        break;
                    }
                    if((wkstr[i]==' ') ||(wkstr[i]=='\t')||
                       (wkstr[i]=='\n')||(wkstr[i]==',')){
                        i++;
                        tmp_delmIndex=i;
                    }
```

```
                    else{
                        i--;
                        break;
                    }
                }
            }
        }
    }


    if((wkstr[i-1]!=' ' ) && (wkstr[i-1]!='\t')&&
       (wkstr[i-1]!='\n') && (wkstr[i-1]!=','))
    {
        memset(wkstr_wd, 0x00, sizeof( wkstr_wd ));
        memcpy(wkstr_wd, &wkstr[tmp_delmIndex], i - tmp_delmIndex);      // 区切り位置を設定する。

        tmpStr = string(wkstr_wd);
        tmpCsvLineData.word.push_back(tmpStr);
        tmpCsvLineData.type.push_back(tmp_delm);
    }

    vCsvLineData.push_back( tmpCsvLineData );

    return true;
}
//-------------------------------------
// １行データの削除
//-------------------------------------
bool CCsvFile::DeleteData(void)
{
    if( p_vCsvLineData != vCsvLineData.end() ){
        if(!vCsvLineData.empty()){
            vCsvLineData.erase(vCsvLineData.begin(), vCsvLineData.end());
            vCsvLineData.clear();
        }
    }
    else return false;

    p_vCsvLineData = vCsvLineData.end();

    return true;
}
//-------------------------------------
// １行データ取得
//-------------------------------------
string CCsvFile::Strings(char delm)
{
    string rtnStr;
    char wkstr[128];
    int i;
    int tmpSize;

    rtnStr = "";
    sprintf( wkstr, "%c", delm );
    if( p_vCsvLineData != vCsvLineData.end() ){
        tmpSize = (int)(p_vCsvLineData->word.size());
        for( i = 0; i < tmpSize; i++ ){
            rtnStr = rtnStr + (p_vCsvLineData->word[i]);
            if(i != tmpSize -1){
                rtnStr = rtnStr + string(wkstr);
            }
        }
    }
    return rtnStr;
}
//-------------------------------------
// １行データ文字列取り出し
//-------------------------------------
string CCsvFile::Strings(int index)
{
    if( p_vCsvLineData != vCsvLineData.end() ){
        if((index<(int)(p_vCsvLineData->word.size())) && (index>=0)){
            return( p_vCsvLineData->word[index]);
        }
    }
    return("");
}
//-------------------------------------
// １行データ文字列取り出し
//-------------------------------------
bool CCsvFile::GetDataStr(string &str, int index )
{
    if( p_vCsvLineData != vCsvLineData.end() ){
        if((index <= (int)(p_vCsvLineData->word.size())) && (index>0))
            str = p_vCsvLineData->word[index-1];
        else{
            str = "";
            return false;
```

```
        }
    }
    else return false;

    return true;
}
//-------------------------------------
// １行データのカウンタ
//-------------------------------------
int  CCsvFile::GetDataCnt(void)
{
    if(p_vCsvLineData!=vCsvLineData.end()) return((int)(p_vCsvLineData->word.size()));

    else return(0);
}
```

```
#ifndef __CCsvFile__
#define __CCsvFile__

#include    <stdio.h>
#include    <stdlib.h>
#include    <string>
#include    <vector>
using namespace std;

//-------------------------------------------------------------
// １行のデータ保存用構造体
//-------------------------------------------------------------
typedef struct stCsvLineData{
    vector<string>  word;
    vector<int>     type;
}stCsvLineData;

//-------------------------------------------------------------
// CSVデータのクラス宣言
//-------------------------------------------------------------
class CCsvFile
{

public:
    CCsvFile();                                         // コンストラクタ
    virtual ~CCsvFile();                                // デストラクタ

    bool ReadFile(string filename);                     // ファイル読み込み
    bool WriteFile(string filename, char delm=' '); // ファイル書き込み
    int  RecSize(void);                                 // レコード数
    bool SetAtRec(int index );                          // レコードのポインタ移動

    bool SetDataStr(string str);                        // １行データ設定
    bool DeleteData(void);                              // １行データ削除
    string Strings(char delm= ' ');                     // １行データ取得
    string Strings(int index);                          // １行データ文字列取り出し
    bool GetDataStr(string &str, int index );           // １行データ文字列取り出し
    int  GetDataCnt(void);                              // １行データのカウンタ

protected:
    void DataClear(void);                               // メモリ内容クリア処理

    vector<stCsvLineData> vCsvLineData;                 // 読み込み及び格納データ
    vector<stCsvLineData>::iterator p_vCsvLineData;     // 読み込み及び格納参照ポインタ
};

#endif
```

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <vector>
#include <string>
#include <algorithm>
#include <iostream>
#include <cstdio>
#include "mapout.h"
#include "CCsvFile.h"
using namespace std;
//-------------------------------------------------------------
// this data is static data
//-------------------------------------------------------------
static stqcurve ftqc;
static sfcmap    fcm;
static sgridmap gfcm;

vector<double> m_vd1rev;
vector<double> m_vd1trq;

double maxtqlin( stqcurve tqc, double rev );
// ---------------------------------------------------------------------------
// uid   : unit ID
// mapname : ファイル名
// rev   : 回転数配列
// tq    : トルク配列
// fc    : 燃料消費量配列
// nfc   : 読み込みデータ数(アイドリングを除くマップ用)
// fcidle : アイドリング時消費量
void readmapfc( int uid, string mapname, struct sfcmap &fcm, int &rtnval )
{
    double idlev, idlet;
    int     flg;
    int     gap, i, j, k;
    double  r, t, f;
    int     head;
    int     ct;
    CCsvFile *pStrList;
    bool    bRet;
    vector<double>::iterator p_tmp1;
    vector<double>::iterator p_tmp2;

    rtnval = 0;

    pStrList = new CCsvFile();
    bRet = pStrList->ReadFile( mapname.c_str() );
    if( bRet == false ){
        fprintf( stderr, "Error : annot open map file :,%s\n", mapname.c_str() );
        rtnval = -1;
        return;
    }
    if( pStrList->RecSize() <= 4 ){
        fprintf( stderr, "Error : faild to read map data. Data No. = %d in %s\n",
                 fcm.ndata, mapname.c_str() );
        rtnval = -2;
        return;
    }
    pStrList->SetAtRec( 3 );

    idlev = (double)atof( pStrList->Strings(0).c_str() );
    idlet = (double)atof( pStrList->Strings(1).c_str() );
    fcm.fcidle = (double)atof( pStrList->Strings(2).c_str() );

    // allocation
    fcm.rev.resize( pStrList->RecSize() -3 );
    fcm.tq.resize(  pStrList->RecSize() -3 );
    fcm.fc.resize(  pStrList->RecSize() -3 );

    fcm.ndata = pStrList->RecSize() -3;
    flg = 0;
    for( i = 4; i <= pStrList->RecSize(); i++ ){
        pStrList->SetAtRec(i);
        fcm.rev[i-4] = (double)( atof( pStrList->Strings(0).c_str() ));
        fcm.tq[i-4] = (double)(  atof( pStrList->Strings(1).c_str() ));
        fcm.fc[i-4] = (double)(  atof( pStrList->Strings(2).c_str() ));
    }

    // SORT MAP DATA WIT_H ENGINE SPEED
    gap = (fcm.ndata+1) /2;
    while( gap > 0 ){
        i = gap;
        while( i <= fcm.ndata ){
            j = i - gap;
            while( ( j >= 1 )&&( fcm.rev[j-1] > fcm.rev[j+gap-1] )){
                p_tmp1 = fcm.rev.begin() + (j-1);
                p_tmp2 = fcm.rev.begin() + (j+gap-1);
```

```
                swap_ranges( p_tmp1, p_tmp1+1, p_tmp2 );

                p_tmp1 = fcm.tq.begin() + (j-1);
                p_tmp2 = fcm.tq.begin() + (j+gap-1);
                swap_ranges( p_tmp1, p_tmp1+1, p_tmp2 );

                p_tmp1 = fcm.fc.begin() + (j-1);
                p_tmp2 = fcm.fc.begin() + (j+gap-1);
                swap_ranges( p_tmp1, p_tmp1+1, p_tmp2 );

                j = j - gap;
            }
            i++;
        }
        gap = gap /2;
    }

    // SORT WITH ENGIN TORQUE
    head = 1;
    r = fcm.rev[head-1];
    ct = 0;
    for( i = 1; i <= fcm.ndata; i++ ){
        ct = ct +1;
        if( ( r != fcm.rev[i-1] )||( i == fcm.ndata ) ){
            if( r != fcm.rev[i-1] ){
                ct = ct-1;
            }
            gap = ( ct + 1 ) /2;
            while( gap > 0 ){
                k = gap;
                while( k <= ct -1 ){
                    j = head + k - gap;
                    while( ( j >= head )&&( fcm.tq[j-1] > fcm.tq[j+gap-1] ) ){
                        p_tmp1 = fcm.rev.begin() + (j-1);
                        p_tmp2 = fcm.rev.begin() + (j+gap-1);
                        swap_ranges( p_tmp1, p_tmp1+1, p_tmp2 );

                        p_tmp1 = fcm.tq.begin() + (j-1);
                        p_tmp2 = fcm.tq.begin() + (j+gap-1);
                        swap_ranges( p_tmp1, p_tmp1+1, p_tmp2 );

                        p_tmp1 = fcm.fc.begin() + (j-1);
                        p_tmp2 = fcm.fc.begin() + (j+gap-1);
                        swap_ranges( p_tmp1, p_tmp1+1, p_tmp2 );

                        j = j - gap;
                    }
                    k = k +1;
                }
                gap = gap / 2;
            }
            r = fcm.rev[i-1];
            ct = 1;
            head = i;
        }
    }

    delete pStrList;
    return;

}

// ********************************************************************************
// SUBROUTINE readtqfc     : readout maximum torque data
// ********************************************************************************
void readtqfc( int uid, string fname, stqcurve &tqc, int &rtnval )
{

    string tmp;
    int i, j, flg, gap, ios;
    CCsvFile *pStrList;
    bool    bRet;
    double revt, tmaxt;
    vector<double>::iterator p_tmp1;
    vector<double>::iterator p_tmp2;
    sttrq       srctrq;
    sttrq       dsttrq;

    tqc.ndata = 0;
    rtnval = 0;

    pStrList = new CCsvFile();
    bRet = pStrList->ReadFile( fname.c_str() );
    if( bRet == false ){
        fprintf( stderr, "Error : annot open map file :,%s\n", fname.c_str() );
        rtnval = -1;
        return;
    }
```

```
    if( pStrList->RecSize() <= 2 ){
        fprintf( stderr, "Error : faild to read map data. Data No. = %d in %s\n",
                 tqc.ndata+1, fname.c_str() );
        rtnval = -2;
        return;
    }
    pStrList->SetAtRec( 3 );

    tqc.ndata = pStrList->RecSize() -2;
    // allocation
    tqc.tq.resize( pStrList->RecSize() -2 );
    tqc.rev.resize( pStrList->RecSize() -2 );

    for( i = 3; i <= pStrList->RecSize(); i++ ){
        pStrList->SetAtRec(i);
        tqc.rev[i-3] = (double)( atof( pStrList->Strings(0).c_str() ) );
        tqc.tq[i-3] = (double)(  atof( pStrList->Strings(1).c_str() ) );
    }

    // sort torque data in ascending order
    gap = (tqc.ndata + 1)/2;
    while( gap != 0 ){
        i = gap;
        while( i <= tqc.ndata ){
            j = i - gap;
            while( ( j >= 1 )&&( tqc.rev[j-1] > tqc.rev[j+gap-1] ) ){
                p_tmp1 = tqc.rev.begin() + (j-1);
                p_tmp2 = tqc.rev.begin() + (j+gap-1);
                swap_ranges( p_tmp1, p_tmp1+1, p_tmp2 );

                p_tmp1 = tqc.tq.begin() + (j-1);
                p_tmp2 = tqc.tq.begin() + (j+gap-1);
                swap_ranges( p_tmp1, p_tmp1+1, p_tmp2 );

                j = j - gap;
            }
            i++;
        }
        gap = gap /2;
    }

    delete pStrList;
    return;

}
void tqcnv( sttrq &src, sttrq &dst )
{
    double nemin, nemax;
    double netmp;
    int rn;
    vector<double>::iterator p_pt;
    vector<double> vd1;
    int i,j, ierr;
    bool skip;

    skip = true;
    if( dst.rev.empty() != true ){
        dst.rev.erase( dst.rev.begin(), dst.rev.end() );
        dst.rev.clear();
    }
    if( dst.tq.empty() != true ){
        dst.tq.erase( dst.tq.begin(), dst.tq.end() );
        dst.tq.clear();
    }

    p_pt = min_element( src.rev.begin(), src.rev.end() );
    nemin = *p_pt;
    p_pt = max_element( src.rev.begin(), src.rev.end() );
    nemax = *p_pt;

    netmp = nemin;
    while( netmp <= nemax ){
        dst.rev.push_back( netmp );
        netmp = netmp +8;
    }
    dst.tq.resize( dst.rev.size() );


    vd1.resize( src.rev.size() );

    dpchimC( src.rev.size(), src.rev, src.tq, vd1, 0, 0, 1, ierr );     // 一定トルク値における回転数-燃料消費量曲線の微係
数を計算.

    dpchfeC( src.rev.size(), src.rev, src.tq, vd1, 0, 0, 1, skip,
                        dst.rev.size(), dst.rev, dst.tq, ierr );
```

```
}
//*****************************************************************************************
// fcgrid4
// make NX x NY grid fuel map
//
// gfcm   : type(sgridmap), グリッド上に回帰した燃費マップ
// fcm    : type(sfcmap), 燃費マップ入力値（測定値）
// ftqc   : type(stqcurve), 摩擦トルク
// retval : integer, 戻り値
// nx,ny  : グリッドデータ点数
//*****************************************************************************************
void fcgrid4( sgridmap &gfcm, sfcmap &fcm, stqcurve &ftqc,
              int &retval, int &nx, int &ny )
{

    int i, j, ierr;
    int rn, tn, tnmax;
    vector<double> r, x;
    vector< vector<double> > t, f;
    vector<int> tqn;
    double nemin, nemax, tqmin, tqmax;
    double dne, dtq;
    double rr;
    bool skip;
    vector< vector<double> > d1, f1;
    vector<double> pchval;
    vector<double>::iterator p_pt;
    vector<double> vf, vd1;


    // ---- グリッドにした燃費マップのXY各方向の範囲を設定 ----
    gfcm.rev.resize( nx );
    gfcm.tq.resize( ny );
    gfcm.fc.resize( nx );
    for( i = 0; i < 20; i++ ){
        gfcm.fc[i].resize( ny );
    }
    gfcm.nx = nx;
    gfcm.ny = ny;
    gfcm.fcidle = fcm.fcidle;

    // ---- グリッドのXY各方向の範囲を設定 ----
    p_pt = min_element( fcm.rev.begin(), fcm.rev.end() );
    nemin = *p_pt;
    p_pt = max_element( fcm.rev.begin(), fcm.rev.end() );
    nemax = *p_pt;
    p_pt = min_element( fcm.tq.begin(), fcm.tq.end() );
    tqmin = *p_pt;
    p_pt = max_element( fcm.tq.begin(), fcm.tq.end() );
    tqmax = *p_pt;

    // ---- 入力マップの回転数方向・トルク方向のデータ数を数える ----
    rn = 1;
    tn = 0;
    tn++;
    tnmax = 1;
    rr = fcm.rev[0];
    for( i = 1; i <= fcm.ndata; i++ ){
        if( rr == fcm.rev[i-1] ){
            tn++;
        }else{
            rn++;
            rr = fcm.rev[i-1];
            if( tn > tnmax ){
                tnmax = tn;
            }
            tn = 2;
        }
    }

    //---- 実測マップを二次元に展開 ----
    r.resize(rn);                                          // 実測マップの回転数,rn個を記憶
    tqn.resize(rn);                                        // 各回転数でのトルクデータ数を記憶
    t.resize(rn);
    for( i = 0; i < rn; i++ ){
        t[i].resize( tnmax );                              // トルク実測値を記憶(2次元)
    }
    f.resize(rn);
    for( i = 0; i < rn; i++ ){
        f[i].resize(tnmax);                                // 燃費実測値を記憶(2次元)
    }

    j = 1;                                                 // 回転数配列rの要素番号(1～rnまで変化)
    r[j-1] = fcm.rev[1-1];                                 // 回転数を記憶
    tqn[j-1] = 1;                                          // トルク値の最初の要素は摩擦トルクとする
    t[j-1][tqn[j-1]-1] = maxtqlin( ftqc, r[j-1] );         // この回転数での摩擦トルクを計算
    f[j-1][tqn[j-1]-1] = 0.00000000;                           // 燃料消費量は0とする
```

```
for( i = 1; i <= fcm.ndata; i++ ){                          // 全データをチェック
    if( r[j-1] == fcm.rev[i-1] ){
        tqn[j-1] = tqn[j-1] + 1;                            // 回転数が変わらない場合のトルク，消費量記憶位置
    }else{
        j++;                                                // 回転数が変わった場合
        r[j-1] = fcm.rev[i-1];                              // 回転数ラベルを更新
        tqn[j-1] = 1;                                       // トルク値の最初の要素は摩擦トルクとする
        t[j-1][tqn[j-1]-1] = maxtqlin( ftqc, r[j-1] );      // 摩擦トルクを計算
        f[j-1][tqn[j-1]-1] = 0.00000000;                        // 燃料消費量は0とする
        tqmin = min( tqmin, t[j-1][tqn[j-1]-1] );               // トルク最小値（グリッド作成に使用）を更新
        tqn[j-1] = tqn[j-1] +1;                             // トルクデータ数を更新
    }
    t[j-1][tqn[j-1]-1] = fcm.tq[i-1];                           // トルク値を格納
    f[j-1][tqn[j-1]-1] = fcm.fc[i-1];                           // 燃料消費量を格納
}

//---- グリッド生成 ----
dne = ( nemax - nemin ) / ((double)( nx - 1 ));             // 回転数間隔を求める
dtq = ( tqmax - tqmin ) / ((double)( ny - 1 ));             // トルク間隔を求める

for( i = 1; i <= nx; i++ ){
    gfcm.rev[i-1] = nemin + dne * ((double)(i-1));          // グリッドx軸の回転数
}
for( i = 1; i <= ny; i++ ){
    gfcm.tq[i-1] = tqmin + dtq * ((double)(i-1));           // グリッドy軸のトルク
}

//---- グリッドデータ補間 ----
//最初に，回転数は測定値のまま，y軸トルク方向をNY個に補間し，rn × ny のデータをつくる
f1.resize(rn);
for( i = 0; i < rn; i++ ){
    f1[i].resize(ny);                                       // rn×NY補間マップ
}
d1.resize(rn);
for( i = 0; i < rn; i++ ){
    d1[i].resize(tnmax);                                    // 微係数配列(2次元,rn×tnmax要素)
}


for( i = 1; i <= rn; i++ ){                                 // 回転数ごとにNY個の補間データ作成
    x.resize( tqn[i-1] );                                   // 作業用に確保
    for( j = 1; j <= tqn[i-1]; j++ ){
        x[j-1] = t[i-1][j-1];                               // トルク値をコピー
    }

    cnvMemorySet( f, vf );
    cnvMemorySet( d1, vd1 );

    dpchimC( tqn[i-1], x, vf, vd1, i-1, 0,
                       rn, ierr );                          // 回転数r(i)におけるトルク-燃料消費量曲線の微係数を計算

    switch(ierr){
        case -1:
            printf("Error : 燃費マップが作成できませんでした(トルク方向の微係数計算に失敗)\n");
            printf("ierr = %d\n",ierr );
            //deallocate(f1, d1, r, tqn, t, f, x )
            retval = ierr;
            return;
    }

    skip = true;                                            // for slatec library
    pchval.resize(ny);                                      // Hermite補間結果を格納
    dpchfeC( tqn[i-1], x, vf, vd1, i-1, 0, rn, skip, gfcm.ny, gfcm.tq, pchval, ierr );

    cnvMemoryRSet( f, vf );
    cnvMemoryRSet( d1, vd1 );

    switch(ierr){
        case -1:
            printf("Error : 燃費マップが作成できませんでした(トルク方向の燃料消費量補間に失敗)\n");
            printf("ierr = %d\n",ierr );
            //deallocate( f1, d1, r, tqn, t, f, pchval, x )
            retval = ierr;
            return;
    }

    for( j = 1; j <= ny; j++ ){
        f1[i-1][j-1] = pchval[j-1];                         // 補間結果をグリッドデータにコピーする
    }

    if( pchval.empty() != true ){
        pchval.erase( pchval.begin(), pchval.end() );
        pchval.clear();                                         // 作業用配列を解放
    }

    if( x.empty() != true ){
        x.erase( x.begin(), x.end() );
        x.clear();
```

```
    }
}

for( i = 0; i < d1.size(); i++ ){
    if( d1[i].empty() != true ){
        d1[i].erase( d1[i].begin(), d1[i].end() );
        d1[i].clear();
    }
}
if( d1.empty() != true ){
    d1.erase( d1.begin(), d1.end() );
    d1.clear();
}

//次にrn×NYの補間マップを用いて回転数方向をNX個に補間し,NX×NYのデータを作る
d1.resize(rn);
for( i = 0; i < rn; i++ ){
    d1[i].resize(ny);                                   // 微係数配列をNX×NY要素とする
}

vf.erase( vf.begin(), vf.end());
vf.clear();
vd1.erase( vd1.begin(), vd1.end());
vd1.clear();

for( i = 1; i <= ny; i++ ){                             // NY個の同一トルク線上で回転数補間

    cnvMemorySet( f1, vf );
    cnvMemorySet( d1, vd1 );

    dpchimC( rn, r, vf, vd1, 0, rn*(i-1), 1, ierr );    // 一定トルク値における回転数-燃料消費量曲線の微係数を計算.
                                                        // 1とびで格納する
    switch(ierr){
        case -1:
            printf("Error : 燃費マップが作成できませんでした(回転数方向の微係数計算に失敗)\n");
            printf("ierr = %d\n",ierr );
            //deallocate( f1, d1, r, tqn, t, f )
            retval = ierr;
            return;
    }

    skip = true;                                        // for slatec library
    pchval.resize(nx);                                  // Hermite補間結果を格納する1次元配列

    dpchfeC( rn, r, vf, vd1, 0, rn*(i-1), 1, skip, gfcm.nx, gfcm.rev, pchval, ierr );

    cnvMemoryRSet( f1, vf );
    cnvMemoryRSet( d1, vd1 );

    switch(ierr){
        case -1:
            printf("Error : 燃費マップが作成できませんでした(回転数方向の燃料消費量補間に失敗)\n");
            printf("ierr = %d\n",ierr );
            //deallocate( f1, d1, r, tqn, t, f, pchval )
            retval = ierr;
            return;
    }

    for( j = 1; j <= nx; j++ ){
        gfcm.fc[j-1][i-1] = pchval[j-1];                // 補間結果をグリッドデータにコピーする
    }

    if( pchval.empty() != true ){
        pchval.erase( pchval.begin(), pchval.end() );
        pchval.clear();                                     // 作業用配列を解放
    }
}


for( i = 0; i < d1.size(); i++ ){
    if( d1[i].empty() != true ){
        d1[i].erase( d1[i].begin(), d1[i].end() );
        d1[i].clear();
    }
}
if( d1.empty() != true ){
    d1.erase( d1.begin(), d1.end() );
    d1.clear();
}

for( i = 0; i < f1.size(); i++ ){
    if( f1[i].empty() != true ){
        f1[i].erase( f1[i].begin(), f1[i].end() );
        f1[i].clear();
    }
}
if( f1.empty() != true ){
    f1.erase( f1.begin(), f1.end() );
```

```
        f1.clear();
    }

    if( r.empty() != true ){
        r.erase( r.begin(), r.end() );
        r.clear();
    }
    if( tqn.empty() != true ){
        tqn.erase( tqn.begin(), tqn.end() );
        tqn.clear();
    }

    for( i = 0; i < t.size(); i++ ){
        if( t[i].empty() != true ){
            t[i].erase( t[i].begin(), t[i].end() );
            t[i].clear();
        }
    }
    if( t.empty() != true ){
        t.erase( t.begin(), t.end() );
        t.clear();
    }

    for( i = 0; i < f.size(); i++ ){
        if( f[i].empty() != true ){
            f[i].erase( f[i].begin(), f[i].end() );
            f[i].clear();
        }
    }
    if( f.empty() != true ){
        f.erase( f.begin(), f.end() );
        f.clear();
    }

}
// ********************************************************************************************
// * double FUNCTION maxtqlin                                                                 *
// * Calculate maximum torque at specified engine speed                                       *
// * type(stqcurve)::tqc  dataset of maximum torque curve                                     *
// * tqc%rev  : engine speed array, rpm                                                       *
// * tqc%tq   : maximum torque array, Nm                                                      *
// * tqc%ndata: number of data                                                                *
// * rev      : input engine speed, rpm                                                       *
// * maxtqlin : output maximum torque, (Nm)                                                   *
// ********************************************************************************************
double maxtqlin( stqcurve tqc, double rev )
{
    int x;
    double rtnval;

    //===== FIND X THAT SATISFIES R(X)<=REV<R(X+1) ========================================
    if ( rev < tqc.rev[1-1] ){
        x = 1;                                                  // extrapolation, rev<r[1]
    }else if( rev >= tqc.rev[tqc.ndata-1] ){
        x = tqc.ndata -1;                                       // extrapolation, rev>=r[ndata]
    }else{
        for( x = 1; x <= tqc.ndata; x++ ){                      // interpolation
            if( (rev >= tqc.rev[x-1] ) && ( rev < tqc.rev[x+1-1] ) ) break;
        }
    }

    rtnval = tqc.tq[x-1] + ( tqc.tq[x+1-1] - tqc.tq[x-1] ) / ( tqc.rev[x+1-1] - tqc.rev[x-1] ) * ( rev - tqc.rev[x-1] );

    return(rtnval);
}
bool fuelCal(double src_rev, double src_trq, double &fuel)
{
    vector<double> vf, vd1;
    vector< vector<double> > t, f;
    vector< vector<double> > d1, f1;
    vector<double> pchval;
    vector<double> trq;
    vector<double> r;
    vector<double> rev;
    vector<double> trqx;
    int i,j,ierr;
    bool skip;
    bool bRet;
    int  index_rev1, index_rev2;
    int  index_tq1, index_tq2;
    double tmpValue;
    vector<double> tmp_fe;
    vector<int> next(2);
    char wkstr[128];

#ifdef FOR_UND_POINT
    sprintf( wkstr, "%.9lf", src_rev );
    wkstr[strlen(wkstr)-1] = 0x00;
```

```
    src_rev = (double)atof(wkstr);

    sprintf( wkstr, "%.9lf", src_trq );
    wkstr[strlen(wkstr)-1] = 0x00;
    src_trq = (double)atof(wkstr);
#endif

    if( gfcm.rev.empty() == true ){
        return( false );
    }
    if( gfcm.tq.empty() == true ){
        return( false );
    }
    if( gfcm.fc.empty() == true ){
        return( false );
    }

    // 対象となるトルクの位置、回転の位置を把握する。
    bRet = fuelIndexSearch( src_rev, src_trq,
                index_rev1, index_rev2,
                index_tq1, index_tq2 );
    if( bRet == false ){
        return( false );
    }
    if( index_rev2 < 0 ){
        index_rev1 = 1; index_rev2 = 0;
    }
    if( index_tq2 < 0 ){
        index_tq1 = 1;  index_tq2 = 0;
    }
    if( index_rev1 == index_rev2 ){
        if( index_rev1 == gfcm.nx ){
            index_rev1 = gfcm.nx -1;
            index_rev2 = index_rev1 -1;
        }
    }
    if( index_tq1 == index_tq2 ){
        if( index_tq1 == gfcm.ny ){
            index_tq1 = gfcm.ny -1;
            index_tq2 = index_tq1 -1;
        }
    }

    // トルク値をコピー
    if( trq.empty() != true ){
        trq.erase( trq.begin(), trq.end() );
        trq.clear();
    }
    for( i = 0; i < gfcm.ny; i++ ){
        trq.push_back( gfcm.tq[i] );
    }

    // 複数バージョン
    // 燃費をコピー
    f.resize(gfcm.nx);
    for( i = 0; i < gfcm.nx; i++ ){
        for( j = 0; j < gfcm.ny; j++ ){
            f[i].push_back( gfcm.fc[i][j] );
        }
    }

    // 回転数をコピー
    if( r.empty() != true ){
        r.erase( r.begin(), r.end() );
        r.clear();
    }
    for( i = 0; i < gfcm.nx; i++ ){
        r.push_back( gfcm.rev[i] );
    }

    rev.resize(gfcm.nx);
    for( i = 0; i < gfcm.nx; i++ ){
        rev[i]= src_rev;
    }
    trqx.resize(gfcm.ny);
    for( i = 0; i < gfcm.ny; i++ ){
        trqx[i]= src_trq;
    }


    // 複数バージョン
    f1.resize(gfcm.nx);
    for( i = 0; i < gfcm.nx; i++ ){
        f1[i].resize(gfcm.ny);                                  // rn×NY補間マップ
    }
    d1.resize(gfcm.nx);
    for( i = 0; i < gfcm.nx; i++ ){
        d1[i].resize(gfcm.ny);                                  // 微係数配列(2次元,rn×tnmax要素)
```

```
    }

    cnvMemorySet( f, vf );
    if( m_vd1rev.empty() == true ){
        cnvMemorySet( d1, vd1 );
    }

    skip = true;                                                    // for slatec library
    pchval.resize(gfcm.ny);                                         // Hermite補間結果を格納

    // 再設定バージョン２０点モード
    /*       トルク数，トルクデータ，燃費データ，微係数結果，offset，offset，結果更新offset，エラー   */

    if( m_vd1rev.empty() == true ){
        for( i = 1; i <= gfcm.nx; i++ ){
                                                        // NEにおけるトルク-燃料消費量曲線の微係数を計算
        dpchimC( gfcm.ny, r, vf, vd1, 0, gfcm.ny*(i-1), 1, ierr );      // 一定トルク値における回転数-燃料消費量曲線の微係
数を計算.
            switch(ierr){
                case -1:
                    printf("Error : 燃費マップが作成できませんでした(トルク方向の微係数計算に失敗)\n");
                    printf("ierr = %d\n",ierr );
                    return(false);
            }
        }
        copy( vd1.begin(), vd1.end(), back_inserter(m_vd1rev) );
    }else{
        copy( m_vd1rev.begin(), m_vd1rev.end(), back_inserter(vd1) );
    }

    tmp_fe.resize(gfcm.nx);
    for( i = 1; i <= gfcm.nx; i++ ){

        dchfev( r[index_rev2], r[index_rev1],
                    vf[index_rev2+(i-1)*gfcm.nx], vf[index_rev1+(i-1)*gfcm.nx],
                    vd1[index_rev2+(i-1)*gfcm.nx], vd1[index_rev1+(i-1)*gfcm.nx], 1,
                    rev, tmp_fe, next, ierr );
        pchval[i-1] = tmp_fe[0];

        switch(ierr){
            case -1:
                printf("Error : 燃費マップが作成できませんでした(トルク方向の燃料消費量補間に失敗)\n");
                printf("ierr = %d\n",ierr );
                return(false);
        }
    }

    cnvMemoryRSet( f, vf );
    cnvMemoryRSet( d1, vd1 );



    for( j = 1; j <= gfcm.ny; j++ ){
        f1[0][j-1] = pchval[j-1];                           // 補間結果をグリッドデータにコピーする
    }

    if( pchval.empty() != true ){
        pchval.erase( pchval.begin(), pchval.end() );
        pchval.clear();                                             // 作業用配列を解放
    }


    for( i = 0; i < d1.size(); i++ ){
        if( d1[i].empty() != true ){
            d1[i].erase( d1[i].begin(), d1[i].end() );
            d1[i].clear();
        }
    }
    if( d1.empty() != true ){
        d1.erase( d1.begin(), d1.end() );
        d1.clear();
    }

    if( vf.empty() != true ){
        vf.erase( vf.begin(), vf.end());
        vf.clear();
    }
    if( vd1.empty() != true ){
        vd1.erase( vd1.begin(), vd1.end());
        vd1.clear();
    }



    if( f.empty() != true ){
        for( i = 0; i < f.size(); i++ ){
            if( f[i].empty() != true ){
                f[i].erase( f[i].begin(), f[i].end() );
```

```
            f[i].clear();
        }
    }
    f.erase( f.begin(), f.end() );
    f.clear();
}

f.resize(1);
for( i = 0; i < gfcm.ny; i++ ){
    f[0].push_back( f1[0][i] );
}

d1.resize(gfcm.nx);
for( i = 0; i < gfcm.nx; i++ ){
    d1[i].resize(gfcm.ny);                                  // 微係数配列をNX×NY要素とする
}

vf.erase( vf.begin(), vf.end());
vf.clear();
vd1.erase( vd1.begin(), vd1.end());
vd1.clear();

cnvMemorySet( f, vf );
cnvMemorySet( d1, vd1 );


dpchimC( gfcm.nx, trq, vf, vd1, 0, 0, 1, ierr );        // 一定トルク値における回転数-燃料消費量曲線の微係数を計算.
                                                        // 1とびで格納する
switch(ierr){
    case -1:
        printf("Error : 燃費マップが作成できませんでした(回転数方向の微係数計算に失敗)\n");
        printf("ierr = %d\n",ierr );
        //deallocate( f1, d1, r, tqn, t, f )
        return(false);
}

skip = true;                                            // for slatec library
pchval.resize(gfcm.nx);                                 // Hermite補間結果を格納する1次元配列


// 直接バージョン
dchfev( trq[index_tq2], trq[index_tq1],
                vf[index_tq2], vf[index_tq1],
                vd1[index_tq2], vd1[index_tq1], 1,
                trqx, tmp_fe, next, ierr );

switch(ierr){
    case -1:
        printf("Error : 燃費マップが作成できませんでした(回転数方向の燃料消費量補間に失敗)\n");
        printf("ierr = %d\n",ierr );
        return(false);
}


fuel = tmp_fe[0];

if( pchval.empty() != true ){
    pchval.erase( pchval.begin(), pchval.end() );
    pchval.clear();                                     // 作業用配列を解放
}

for( i = 0; i < d1.size(); i++ ){
    if( d1[i].empty() != true ){
        d1[i].erase( d1[i].begin(), d1[i].end() );
        d1[i].clear();
    }
}
if( d1.empty() != true ){
    d1.erase( d1.begin(), d1.end() );
    d1.clear();
}

for( i = 0; i < f.size(); i++ ){
    if( f[i].empty() != true ){
        f[i].erase( f[i].begin(), f[i].end() );
        f[i].clear();
    }
}
if( f.empty() != true ){
    f.erase( f.begin(), f.end() );
    f.clear();
}

if( vf.empty() != true ){
    vf.erase( vf.begin(), vf.end());
    vf.clear();
}
if( vd1.empty() != true ){
```

```
        vd1.erase( vd1.begin(), vd1.end());
        vd1.clear();
    }
    if( tmp_fe.empty() != true ){
        tmp_fe.erase( tmp_fe.begin(), tmp_fe.end() );
        tmp_fe.clear();
    }

}


double fuelCal( double rev_down, double rev_up,
                    double trq_down, double trq_up,
                    double rev_fuel_down, double rev_fuel_up,
                    double trq_fuel_down, double trq_fuel_up,
                    double src_rev, double src_trq )
{
    double s_rev;
    double s_trq;
    double s_fuel_rev;
    double s_fuel_trq;
    double rtnval;

    s_rev = (src_rev - rev_down)/(rev_up - rev_down);
    s_trq = (src_trq - trq_down)/(trq_up - trq_down);
    s_fuel_rev = fabs(rev_fuel_up - rev_fuel_down) * s_rev + rev_fuel_down;
    s_fuel_trq = fabs(rev_fuel_up - trq_fuel_down) * s_trq + trq_fuel_down;


    rtnval = (s_fuel_rev + s_fuel_trq + trq_fuel_down + trq_fuel_up)/4.0;

    return( rtnval );
}
bool fuelInit( char *mapf, char *ftqf, int nx, int ny )
{
    int retval;

    readmapfc( 13, mapf, fcm, retval );
    if( retval != 0 ){
        return( false );
    }

    readtqfc( 14, ftqf, ftqc, retval );
    if( retval != 0 ){
        return( false );
    }

    fcgrid4( gfcm, fcm, ftqc, retval, nx, ny );
    if( retval != 0 ){
        return( false );
    }

    return( true );
}
bool fuelIndexSearch( double rev, double trq,
                int &index_rev1, int &index_rev2,
                int &index_tq1, int &index_tq2 )
{
    if( gfcm.rev.empty() == true ){
        return( false );
    }
    if( gfcm.tq.empty() == true ){
        return( false );
    }

    for( index_rev1 = 0, index_rev2 = 0; index_rev1 <= gfcm.nx-1; index_rev1++, index_rev2++ ){
        if( rev < gfcm.rev[index_rev1] ){
           index_rev2 = index_rev1-1;
           break;
        }
    }

    for( index_tq1 = 0, index_tq2 = 0; index_tq1 <= gfcm.ny-1; index_tq1++, index_tq2++ ){
        if( trq < gfcm.tq[index_tq1] ){
            index_tq2 = index_tq1-1;
            break;
        }
    }

    return( true );

}
bool fuelSearch( double rev, double trq, double &fuel )
{
    bool bRet;
    double rtn_fuel;

    if( gfcm.rev.empty() == true ){
```

```
        return( false );
    }
    if( gfcm.tq.empty() == true ){
        return( false );
    }
    if( gfcm.fc.empty() == true ){
        return( false );
    }



    try{
        if( rev <= 0 ){
            rtn_fuel = gfcm.fcidle;
        }else{
            fuelCal( (double)rev, (double)trq, rtn_fuel );
        }
    }catch(...){
        rtn_fuel = gfcm.fcidle;
    }

    if( rtn_fuel < 0 ){
        fuel = 0.0;
    }else{
        fuel = (double)rtn_fuel;
    }


    return( true );
}
```

```
#ifndef __MAPOUT_H__
#define __MAPOUT_H__
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <vector>
using namespace std;


//#define FOR_UND_POINT   0


typedef struct stqcurve{
    int ndata;
    vector<double> rev;
    vector<double> tq;
}stqcurve;

typedef struct sfcmap{
    double  fcidle;
    int     ndata;
    vector<double>  rev;
    vector<double>  tq;
    vector<double>  fc;
}sfcmap;

typedef struct sgridmap{
    double  fcidle;
    int     nx;
    int     ny;
    vector<double>  rev;
    vector<double>  tq;
    vector< vector<double> >    fc;
}sgridmap;

typedef struct sttrq{
    vector<double> rev;
    vector<double> tq;
}sttrq;

void dchfev( double x1, double x2, double f1, double f2, double d1, double d2,
                int ne, vector<double> &xe, vector<double> &fe, vector<int> &next, int &ierr );
void dpchfeC( int n, vector<double> &x,
                    vector<double> &f,
                    vector<double> &d,
                    int nx, int ny,
                    int incfd, bool skip, int ne,
                    vector<double> &xe,
                    vector<double> &fe,
                    int &ierr );
void dpchimC( int n,
            vector<double> &x,
            vector<double> &f,
            vector<double> &d,
            int nx, int ny,
            int incfd, int &ierr );
double dpchst( double arg1, double arg2 );

bool cnvMemorySet( vector< vector<double> > org, vector<double> & dst );
bool cnvMemoryRSet( vector< vector<double> > &org, vector<double> src );

void tqcnv( sttrq &src, sttrq &dst );

double fuelCal( double rev_down, double rev_up,
                    double trq_down, double trq_up,
                    double rev_fuel_down, double rev_fuel_up,
                    double trq_fuel_down, double trq_fuel_up,
                    double src_rev, double src_trq );
bool fuelInit( char *mapf, char *ftqf, int nx, int ny );
bool fuelIndexSearch( double rev, double trq,
                int &index_rev1, int &index_rev2,
                int &index_tq1, int &index_tq2 );
bool fuelSearch( double rev, double trq, double &fuel );


extern vector<double> m_vd1rev;
extern vector<double> m_vd1trq;

#endif
```

```
#include <string>

#define NB_DIV              25
#define NB_DIV_CATEGORIES   6

std::string divNames[NB_DIV]={
"T1","T2","T3","T4","T5","T6","T7","T8","T9","T10","T11",
"TT1","TT2",
"BR1","BR2","BR3","BR4","BR5",
"B1","B2","B3","B4","B5","B6","B7"};

double divValues[NB_DIV][NB_DIV_CATEGORIES] = {
1.957,    1.49,     3,   1.982,    1.695,    10,
2.356,    2,        3,   2.099,    1.751,    10,
2.652,    2.995,    3,   2.041,    1.729,    10,
2.979,    3.749,    3,   2.363,    2.161,    10,
3.543,    4.275,    2,   2.454,    2.235,    10,
3.659,    5.789,    2,   2.625,    2.239,    10,
4.048,    7.483,    2,   2.541,    2.35,     10,
4.516,    7.992,    2,   2.572,    2.379,    10,
5.533,    8.9,      2,   2.745,    2.48,     10,
8.688,    11.089,   2,   3.049,    2.49,     10,
8.765,    15.53,    2,   2.934,    2.49,     30,
10.525,   24,       2,   2.927,    2.49,     20,
19.028,   40,       2,   2.89,     2.49,     10,
5.186,    0,        39,  2.88,     2.072,    0,
6.672,    0,        46,  2.947,    2.301,    0,
7.324,    0,        62,  2.949,    2.304,    0,
8.654,    0,        77,  2.969,    2.385,    0,
9.79,     0,        79,  2.962,    2.49,     0,
3.543,    0,        29,  2.593,    2.027,    10,
5.622,    0,        29,  3.019,    2.197,    10,
6.608,    0,        49,  3.105,    2.314,    10,
8.022,    0,        58,  3.16,     2.399,    10,
9.774,    0,        60,  3.168,    2.49,     10,
12.11,    0,        62,  3.32,     2.49,     35,
14.583,    0,        51,  3.668,    2.49,     35};
```

```
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include <math.h>
#include <string.h>
#include <vector>
#include "mapout.h"

#ifdef __GNUC__
template <class T> T max( T, T);
template <class T> T min( T, T);
#endif


double dpchst( double arg1, double arg2 ){

    char *str;
    char argstr1[64];
    char argstr2[64];
    char defstr[64];
    int ndig;
    int sign1, dec1;
    int sign2, dec2;
    double tmpVal;

    ndig = 8;
    tmpVal = 0.00000000;

    memset( defstr, 0x00, sizeof( defstr ) );
    memset( argstr1, 0x00, sizeof( argstr1 ) );
    memset( argstr2, 0x00, sizeof( argstr2 ) );

    str = fcvt( tmpVal, ndig, &dec1, &sign1 );
    strcpy( defstr, str );
    str = fcvt( arg1, ndig, &dec1, &sign1 );
    strcpy( argstr1, str );
    str = fcvt( arg2, ndig, &dec2, &sign2 );
    strcpy( argstr2, str );


    if( strcmp( defstr, argstr1 ) == 0 ){
        return(0.00000000);
    }else if( strcmp( defstr, argstr2 ) == 0 ){
        return(0.00000000);
    }else if( sign1 != sign2 ){
    return( -1.00000000 );
    }

    return( 1.00000000 );
}
//****************************************************************************
// dpchim:  Piecewise Cubic Hermite Interpolation to Monotone data.
//****************************************************************************
void dpchimC( int n,
            vector<double> &x,
            vector<double> &f,
            vector<double> &d,
            int nx, int ny,
            int incfd, int &ierr )
{
    int i, nsec;
    double dx1, dx2, del1, del2, dsave, dxsum, dxsumt3, dmax, dmin;
    double w1, w2, drat1, drat2;
    double sgn;
    char wkstr[128];


    ierr = 0;                                                   // エラー指標を初期化

    //---- 引数チェック ----
    if( n < 2 ){                                                // データ点数
        ierr = -1;                                              // n < 2
        return;
    }

    if( incfd < 1 ){                                            // 変数f,dの指標インクリメント時のステップ幅
        ierr = -2;                                              // incfd < 1
        return;
    }

    for( i = 2; i <= n; i++ ){                                  // 配列xがソート済みか,単調増加か確認
        if( x[i-1] > x[i-1-1] ){
            continue;
        }else{
            ierr = -3;                                          //  ソートされていなければエラー
            return;
        }
    }
```

```
    //---- Hermite補間実行 ----
    nsec = n-1;                                                 // データ区間数

    dx1 = x[2-1] - x[1-1];                                      // 下限側データ間隔
    del1 = (f[nx+(ny+1)*incfd] - f[nx+(ny)*incfd]) / dx1;       // 勾配
    dsave = del1;                                               // 記憶

    if( n == 2 ){                                               // データ数が2の場合は直線補間とする
        d[nx+(ny+1-1)*incfd] = del1;                                // 微係数設定
        d[nx+(ny+2-1)*incfd] = del1;                                // 同じ値を設定して終了

#ifdef FOR_UND_POINT
        sprintf( wkstr, "%.9f", d[nx+(ny*incfd+1-1)] );
        wkstr[strlen(wkstr)-1] = 0x00;
        d[nx+(ny*incfd+1-1)] = atof( wkstr );
        sprintf( wkstr, "%.9f", d[nx+(ny*incfd+2-1)] );
        wkstr[strlen(wkstr)-1] = 0x00;
        d[nx+(ny*incfd+2-1)] = atof( wkstr );
#endif
        return;
    }


    dx2 = x[3-1] - x[2-1];                                      // データ数>2の場合,隣の区間幅を計算
    del2 = ( f[nx+(ny+3-1)*incfd] - f[nx+(ny+1)*incfd] ) / dx2;     // 勾配を計算

    dxsum = dx1 + dx2;                                          // 下限側の勾配を決める
    w1 = ( dx1 + dxsum ) / dxsum;
    w2 = -dx1 / dxsum;

    d[nx+(ny)*incfd] = w1 * del1 + w2 * del2;                   // 勾配基準値

#ifdef FOR_UND_POINT
    sprintf( wkstr, "%.9f", d[nx+(ny)*incfd] );
    wkstr[strlen(wkstr)-1] = 0x00;
    d[nx+(ny)*incfd] = atof( wkstr );
#endif
    if( dpchst( (double)d[nx+(ny)*incfd], (double)del1 ) <= 0.00000000 ) {      // 両引数の符号が異なる場合
        d[nx+(ny)*incfd] = 0.00000000;                          // 下限側勾配=0とする
    }else if( dpchst( (double)del1, (double)del2 ) < 0.00000000 ){              // 3点の2区間で勾配符号が異なる場合
        dmax = 3.00000000 * del1;
        if( fabs( d[nx+(ny)*incfd] ) > fabs( dmax ) ) {     // 微係数はdel1の3倍を上限として補正
            d[nx+(ny)*incfd] = dmax;

#ifdef FOR_UND_POINT
            sprintf( wkstr, "%.9f", d[nx+(ny)*incfd] );
            wkstr[strlen(wkstr)-1] = 0x00;
            d[nx+(ny)*incfd] = atof( wkstr );
#endif
        }
    }

    for( i = 2; i <= nsec; i++ ){                               // 内部区間の勾配を決める
        if( i != 2 ){                                           // i=2についてはループ前に処理済みなのでスキップ
            dx1 = dx2;                                          // 3点の前方区間幅
            dx2 = x[i+1-1] - x[i-1];                            // 3点の後方区間の増分
            dxsum = dx1 + dx2;                                  // 3点の区間幅
            del1 = del2;                                        // 勾配をコピー
            del2 = ( f[nx+(ny+i+1-1)*incfd] - f[nx+(ny+i-1)*incfd] )
                                / dx2;                          // 新規分勾配を計算
        }

        d[nx+(ny+i-1)*incfd] = 0.00000000;                          // 微係数の初期値0
        sgn = dpchst( (double)del1, (double)del2 );                             // 勾配の符号比較

        if( sgn == -1.000000000 ){                              // 符号が逆の場合
            ierr = ierr + 1;                                    // 変曲点の数を数える
            dsave = del2;                                       // 記憶
        }else if( sgn == 0.00000000 ){                          // 勾配0の区間がある場合
            if( del2 != 0.00000000 ){                           // del1=0の時,del2をdsaveに記憶
                if( dpchst( (double)dsave, (double)del2 ) < 0.00000000 ){
                    ierr = ierr+1;                              // 変曲点の数を数える
                }
                dsave = del2;
            }
        }else{                                                  // 勾配符号が同じ場合
            dxsumt3 = dxsum * 3.00000000;
            w1 = (dxsum + dx1) / dxsumt3;
            w2 = (dxsum + dx2) / dxsumt3;
            dmax = max( fabs(del1), fabs(del2) );
            dmin = min( fabs(del1), fabs(del2) );
            drat1 = del1 / dmax;
            drat2 = del2 / dmax;
            d[nx+(ny+i-1)*incfd] = dmin /
                        ( w1 * drat1 + w2 * drat2 );            // 微係数を計算
#ifdef FOR_UND_POINT
            sprintf( wkstr, "%.9f", d[nx+(ny+i-1)*incfd] );
```

```
            wkstr[strlen(wkstr)-1] = 0x00;
            d[nx+(ny+i-1)*incfd] = atof( wkstr );
#endif
        }
    }

    w1 = -dx2 / dxsum;                                          // 区間上限側の端点処理
    w2 = (dx2 + dxsum ) / dxsum;
    d[nx+(ny+n-1)*incfd] = w1 * del1 + w2 * del2;                   // 微係数の初期値を設定

#ifdef FOR_UND_POINT
    sprintf( wkstr, "%.9f", d[nx+(ny+n-1)*incfd] );
    wkstr[strlen(wkstr)-1] = 0x00;
    d[nx+(ny+n-1)*incfd] = atof( wkstr );
#endif

    if( dpchst( (double)d[nx+(ny+n-1)*incfd], (double)del2 ) <= 0.00000000 ){       // 勾配符号により値を補正
        d[nx+(ny+n-1)*incfd] = 0.00000000;

#ifdef FOR_UND_POINT
        sprintf( wkstr, "%.9f", d[nx+(ny+n-1)*incfd] );
        wkstr[strlen(wkstr)-1] = 0x00;
        d[nx+(ny+n-1)*incfd] = atof( wkstr );
#endif
    }else if( dpchst( (double)del1, (double)del2 ) < 0.00000000 ){
        dmax = 3.00000000 * del2;
        if( fabs( d[nx+(ny+n-1)*incfd] ) > fabs( dmax ) ){
            d[nx+(ny+n-1)*incfd] = dmax;

#ifdef FOR_UND_POINT
            sprintf( wkstr, "%.9f", d[nx+(ny+n-1)*incfd] );
            wkstr[strlen(wkstr)-1] = 0x00;
            d[nx+(ny+n-1)*incfd] = atof( wkstr );
#endif
        }
    }

    return;
}


void dpchfeC( int n, vector<double> &x,
                     vector<double> &f,
                     vector<double> &d,
                     int nx, int ny,
                     int incfd, bool skip, int ne,
                     vector<double> &xe,
                     vector<double> &fe,
                     int &ierr )
{

    int i, ierc, ir, j, j2, jfirst, nj;
    vector<int> next(2);
    vector<double> tmp_xe;
    vector<double> tmp_fe;

    ierr = 0;

    //---- 引数チェック ----
    if( skip ){
        if( n < 2 ){
            ierr = -1;                                          // データ点数 < 2
            return;
        }

        if( incfd < 1 ){
            ierr = -2;                                          // 変数f,dの指標インクリメント時のステップ幅
            return;
        }
        for( i = 2; i <= n; i++ ){                              // 配列xがソート済みか,単調増加か確認
            if( x[i-1] <= x[i-1-1] ){
                ierr = -3;                                      // 単調増加ではない
                return;
            }
        }

        if( ne < 1 ){
            ierr = -4;                                          // number of evaluation point <1
            return;
        }
    }


    skip = true;
    jfirst = 1;
    ir = 2;

    while( jfirst <= ne ){
```

```
        j2 = ne + 1;
        for( j = jfirst; j != ne+1; j++ ){
            if( xe[j-1] >= x[ir-1] ){
                if( ir == n ){
                    j2 = ne +1;
                }else{
                    j2 = j;
                }
                break;
            }
        }
        nj = j2 - jfirst;

        if( nj != 0 ){
            tmp_xe.clear();
            tmp_fe.clear();
            copy( xe.begin() + jfirst -1, xe.end(), back_inserter(tmp_xe) );
            copy( fe.begin() + jfirst -1, fe.end(), back_inserter(tmp_fe) );

            dchfev( x[ir-1-1], x[ir-1],
                    f[nx+(ny+ir-1-1)*incfd], f[nx+(ny+ir-1)*incfd],
                    d[nx+(ny+ir-1-1)*incfd], d[nx+(ny+ir-1)*incfd], nj,
                    tmp_xe, tmp_fe, next, ierc );
            if( ierc < 0 ){
                ierr = -5;
                return;
            }
            copy( tmp_xe.begin(), tmp_xe.end(), xe.begin() + jfirst -1 );
            copy( tmp_fe.begin(), tmp_fe.end(), fe.begin() + jfirst -1 );

            if( next[2-1] == 0 ){
            }else{
                if( ir < n ){
                    ierr = -5;
                    return;
                }else{
                    ierr = ierr + next[2-1];
                }
            }

            if( next[1-1] == 0 ){
            }else{
                if( ir > 2 ){
                    for( i = jfirst; i != j2-1+1; i++ ){
                        if( xe[i-1] < x[ir-1-1] ){
                            break;
                        }
                    }
                    if( i > j2-1 ){
                        ierr = -5;
                        return;
                    }
                    j2 = i;
                    for( i = 1; i != ir-1+1; i++ ){
                        if( xe[j2-1] < x[i-1] ){
                            break;
                        }
                    }
                    ir = max( 1, i-1 );
                }else{
                    ierr = ierr + next[1-1];
                }
            }
            jfirst = j;
        }

        ir = ir +1;
        if( ir > n ){
            break;
        }
    }
}
void dchfev( double x1, double x2, double f1, double f2, double d1, double d2,
                int ne, vector<double> &xe, vector<double> &fe, vector<int> &next, int &ierr )
{

    int i;
    double c2, c3, del1, del2, delta, h, x, xmi, xma;
    char wkstr[128];

    ierr = 0;

    //---- 引数チェック ----
    if( ne < 1 ){
        ierr = -1;                                          // number of evaluation  points < 1
        return;
    }
```

```
    h = x2 - x1;
    if( h == 0.00000000 ){
        ierr = -2;                                          // 区間幅=0の場合計算不能となる
        return;
    }

    next[1-1] = 0;
    next[2-1] = 0;
    xmi = min( 0.00000000, h );
    xma = max( 0.00000000, h );

    delta = (f2 - f1) /h;
    del1 = (d1 - delta) /h;
    del2 = (d2 - delta) /h;
    c2 = -(del1 + del1 + del2 );
    c3 = (del1 + del2 ) /h;

    for( i = 1; i <= ne; i++ ){
        x = xe[i-1] - x1;
        fe[i-1] = (double)(f1 + x * (d1 + x * (c2 + x * c3 )));

#ifdef FOR_UND_POINT
        sprintf( wkstr, "%.9lf", fe[i-1] );
        wkstr[strlen(wkstr)-1] = 0x00;
        fe[i-1] = atof(wkstr);
#endif
        if( fabs(fe[i-1]) < 0.000000001 ){
            fe[i-1] = 0.00000000;
        }
        if( x < xmi ){
            next[1-1] = next[1-1] + 1;
        }
        if( x > xma ){
            next[2-1] = next[2-1] + 1;
        }
    }

    return;
}

bool cnvMemorySet( vector< vector<double> > org, vector<double> & dst )
{
    int i,j;
    int rn;

    rn = org.size();
    for( i = 0; i < org.size(); i++ ){
        rn = rn + org[i].size();
    }

    if( dst.empty() == true ){
        dst.resize(rn);
    }

    for( i = 0; i < org.size(); i++ ){
        for( j = 0; j < org[i].size(); j++ ){
            rn = j*org.size()+i;
            dst[rn] = org[i][j];
        }
    }

    return(true);
}
bool cnvMemoryRSet( vector< vector<double> > &org, vector<double> src )
{
    int i,j;
    int rn;

    for( i = 0; i < org.size(); i++ ){
        for( j = 0; j < org[i].size(); j++ ){
            rn = j*org.size()+i;
            org[i][j] = src[rn];
        }
    }

    return( true );
}
```

```
#define MODE1_SIZE  1830
double mode1[MODE1_SIZE][3] = {
0,  0,  0,
1,  0,  0,
2,  0,  0,
3,  0,  0,
4,  0,  0,
5,  0,  0,
6,  0,  0,
7,  0,  0,
8,  0,  0,
9,  0,  0,
10, 0,  0,
11, 0,  0,
12, 0,  0,
13, 0,  0,
14, 0,  0,
15, 0,  0,
16, 0,  0,
17, 0,  0,
18, 0,  0,
19, 0,  0,
20, 0,  0,
21, 0,  0,
22, 0,  0,
23, 0,  0,
24, 0,  0,
25, 4.19,   0,
26, 8.32,   0,
27, 12.33,  0,
28, 16.05,  0,
29, 18.74,  0,
30, 20.28,  0,
31, 21.48,  0,
32, 23.13,  0,
33, 25.17,  0,
34, 27.19,  0,
35, 28.97,  0,
36, 30.43,  0,
37, 31.46,  0,
38, 32.24,  0,
39, 33.16,  0,
40, 34.29,  0,
41, 35.4,   0,
42, 36.57,  0,
43, 38.08,  0,
44, 39.65,  0,
45, 40.59,  0,
46, 40.87,  0,
47, 41.03,  0,
48, 41.23,  0,
49, 41.24,  0,
50, 41.15,  0,
51, 41.11,  0,
52, 41.02,  0,
53, 40.97,  0,
54, 41.25,  0,
55, 41.78,  0,
56, 42.2,   0,
57, 42.54,  0,
58, 42.96,  0,
59, 43.37,  0,
60, 43.84,  0,
61, 44.73,  0,
62, 46.1,   0,
63, 47.57,  0,
64, 48.85,  0,
65, 49.89,  0,
66, 50.56,  0,
67, 50.81,  0,
68, 50.84,  0,
69, 50.87,  0,
70, 50.88,  0,
71, 50.71,  0,
72, 50.31,  0,
73, 49.79,  0,
74, 49.16,  0,
75, 48.09,  0,
76, 46.37,  0,
77, 44.14,  0,
78, 41.46,  0,
79, 38.22,  0,
80, 34.76,  0,
81, 31.55,  0,
82, 28.16,  0,
83, 23.82,  0,
84, 18.88,  0,
85, 14.51,  0,
```

```
86, 11.13,  0,
87, 8.59,   0,
88, 7.36,   0,
89, 8.01,   0,
90, 9.99,   0,
91, 12.29,  0,
92, 14.48,  0,
93, 16.35,  0,
94, 17.11,  0,
95, 15.78,  0,
96, 12.39,  0,
97, 7.15,   0,
98, 1.8,    0,
99, 0,  0,
100,    0,  0,
101,    0,  0,
102,    0,  0,
103,    0,  0,
104,    0,  0,
105,    0,  0,
106,    0,  0,
107,    0,  0,
108,    0,  0,
109,    0,  0,
110,    0,  0,
111,    0,  0,
112,    0,  0,
113,    0,  0,
114,    0,  0,
115,    0,  0,
116,    0,  0,
117,    0,  0,
118,    0,  0,
119,    0,  0,
120,    0,  0,
121,    0,  0,
122,    0,  0,
123,    0,  0,
124,    0,  0,
125,    0,  0,
126,    0,  0,
127,    0,  0,
128,    0,  0,
129,    0,  0,
130,    0,  0,
131,    0,  0,
132,    0,  0,
133,    0,  0,
134,    0,  0,
135,    0,  0,
136,    0,  0,
137,    0,  0,
138,    0,  0,
139,    0,  0,
140,    0,  0,
141,    0,  0,
142,    0,  0,
143,    0,  0,
144,    0,  0,
145,    0,  0,
146,    0,  0,
147,    0,  0,
148,    0,  0,
149,    0,  0,
150,    0,  0,
151,    0,  0,
152,    0,  0,
153,    0,  0,
154,    0,  0,
155,    0,  0,
156,    0,  0,
157,    0,  0,
158,    0,  0,
159,    0,  0,
160,    0,  0,
161,    0,  0,
162,    0,  0,
163,    3.7,    0,
164,    8.97,   0,
165,    10.99,  0,
166,    11.48,  0,
167,    15.12,  0,
168,    20.34,  0,
169,    23.32,  0,
170,    25.11,  0,
171,    27.74,  0,
172,    30.38,  0,
173,    32.93,  0,
```

```
174,    36.44,  0,
175,    39.59,  0,
176,    40.72,  0,
177,    41.41,  0,
178,    43.5,   0,
179,    44.4,   0,
180,    45.24,  0,
181,    45.41,  0,
182,    45.17,  0,
183,    44.76,  0,
184,    44.36,  0,
185,    44.01,  0,
186,    43.54,  0,
187,    42.85,  0,
188,    42.35,  0,
189,    42.47,  0,
190,    42.94,  0,
191,    43.2,   0,
192,    43.31,  0,
193,    43.57,  0,
194,    43.96,  0,
195,    44.49,  0,
196,    45.41,  0,
197,    46.55,  0,
198,    47.53,  0,
199,    48.52,  0,
200,    49.86,  0,
201,    51.32,  0,
202,    52.56,  0,
203,    53.69,  0,
204,    54.81,  0,
205,    55.85,  0,
206,    56.88,  0,
207,    57.88,  0,
208,    58.67,  0,
209,    59.31,  0,
210,    59.92,  0,
211,    60.14,  0,
212,    59.88,  0,
213,    59.7,   0,
214,    59.85,  0,
215,    59.86,  0,
216,    59.62,  0,
217,    59.59,  0,
218,    59.81,  0,
219,    59.79,  0,
220,    59.49,  0,
221,    59.24,  0,
222,    59.05,  0,
223,    58.78,  0,
224,    58.53,  0,
225,    58.37,  0,
226,    58.22,  0,
227,    58.08,  0,
228,    58.06,  0,
229,    58.09,  0,
230,    58.05,  0,
231,    57.89,  0,
232,    57.72,  0,
233,    57.61,  0,
234,    57.52,  0,
235,    57.37,  0,
236,    57.14,  0,
237,    56.8,   0,
238,    56.53,  0,
239,    56.71,  0,
240,    57.39,  0,
241,    57.96,  0,
242,    57.98,  0,
243,    57.78,  0,
244,    57.82,  0,
245,    58.01,  0,
246,    58.06,  0,
247,    57.8,   0,
248,    56.98,  0,
249,    55.49,  0,
250,    53.69,  0,
251,    51.95,  0,
252,    50.25,  0,
253,    48.7,   0,
254,    47.64,  0,
255,    47.06,  0,
256,    46.64,  0,
257,    46.3,   0,
258,    46.39,  0,
259,    47.18,  0,
260,    48.55,  0,
261,    49.91,  0,
```

```
262,    50.85,  0,
263,    51.65,  0,
264,    52.81,  0,
265,    54.13,  0,
266,    55.1,   0,
267,    55.75,  0,
268,    56.29,  0,
269,    56.14,  0,
270,    54.54,  0,
271,    51.61,  0,
272,    48.27,  0,
273,    45.4,   0,
274,    43.49,  0,
275,    42.66,  0,
276,    42.71,  0,
277,    43.29,  0,
278,    44.16,  0,
279,    45.28,  0,
280,    46.64,  0,
281,    48.05,  0,
282,    49.42,  0,
283,    51.05,  0,
284,    52.97,  0,
285,    54.57,  0,
286,    55.57,  0,
287,    56.53,  0,
288,    57.67,  0,
289,    58.42,  0,
290,    58.81,  0,
291,    59.56,  0,
292,    60.52,  0,
293,    60.89,  0,
294,    60.87,  0,
295,    61.27,  0,
296,    61.88,  0,
297,    62.11,  0,
298,    62.23,  0,
299,    62.39,  0,
300,    61.87,  0,
301,    60.48,  0,
302,    59.06,  0,
303,    58.16,  0,
304,    57.46,  0,
305,    56.79,  0,
306,    56.36,  0,
307,    56.16,  0,
308,    56.09,  0,
309,    56.15,  0,
310,    56.18,  0,
311,    56, 0,
312,    55.71,  0,
313,    55.6,   0,
314,    55.76,  0,
315,    56.26,  0,
316,    57.22,  0,
317,    58.37,  0,
318,    59.12,  0,
319,    59.37,  0,
320,    59.53,  0,
321,    59.73,  0,
322,    59.74,  0,
323,    59.59,  0,
324,    59.56,  0,
325,    59.65,  0,
326,    59.86,  0,
327,    60.4,   0,
328,    61.23,  0,
329,    61.99,  0,
330,    62.64,  0,
331,    63.32,  0,
332,    63.74,  0,
333,    63.61,  0,
334,    63.25,  0,
335,    62.88,  0,
336,    62.25,  0,
337,    61.48,  0,
338,    61.06,  0,
339,    60.78,  0,
340,    60, 0,
341,    58.97,  0,
342,    58.32,  0,
343,    58.01,  0,
344,    57.65,  0,
345,    57.2,   0,
346,    56.65,  0,
347,    55.92,  0,
348,    55.27,  0,
349,    54.77,  0,
```

```
350,    54.16,  0,
351,    53.49,  0,
352,    53.06,  0,
353,    52.74,  0,
354,    52.38,  0,
355,    52.25,  0,
356,    52.33,  0,
357,    52.21,  0,
358,    52.05,  0,
359,    52.32,  0,
360,    52.64,  0,
361,    52.38,  0,
362,    51.61,  0,
363,    50.48,  0,
364,    48.76,  0,
365,    46.68,  0,
366,    44.77,  0,
367,    42.88,  0,
368,    40.6,   0,
369,    38.17,  0,
370,    35.7,   0,
371,    32.76,  0,
372,    28.21,  0,
373,    23.82,  0,
374,    20.17,  0,
375,    16.37,  0,
376,    10.92,  0,
377,    4.99,   0,
378,    1.06,   0,
379,    0,  0,
380,    0,  0,
381,    0,  0,
382,    1.78,   0,
383,    4.02,   0,
384,    7.51,   0,
385,    12.17,  0,
386,    16.29,  0,
387,    18.22,  0,
388,    19.22,  0,
389,    21.99,  0,
390,    24.7,   0,
391,    26.87,  0,
392,    27.96,  0,
393,    28.32,  0,
394,    28.05,  0,
395,    27.45,  0,
396,    27.05,  0,
397,    26.82,  0,
398,    26.53,  0,
399,    26.69,  0,
400,    27.8,   0,
401,    29.17,  0,
402,    29.87,  0,
403,    30.11,  0,
404,    30.63,  0,
405,    31.59,  0,
406,    32.84,  0,
407,    34.17,  0,
408,    35.18,  0,
409,    35.58,  0,
410,    35.67,  0,
411,    36.07,  0,
412,    37.08,  0,
413,    38.37,  0,
414,    39.26,  0,
415,    39.6,   0,
416,    39.96,  0,
417,    40.58,  0,
418,    40.91,  0,
419,    40.73,  0,
420,    40.53,  0,
421,    40.51,  0,
422,    40.37,  0,
423,    40.06,  0,
424,    39.76,  0,
425,    39.46,  0,
426,    39.41,  0,
427,    39.81,  0,
428,    39.89,  0,
429,    38.96,  0,
430,    37.88,  0,
431,    37.95,  0,
432,    39.17,  0,
433,    40.68,  0,
434,    41.98,  0,
435,    43.09,  0,
436,    44.24,  0,
437,    45.66,  0,
```

```
438,    47.17,  0,
439,    48.25,  0,
440,    48.61,  0,
441,    48.39,  0,
442,    47.83,  0,
443,    47.28,  0,
444,    46.95,  0,
445,    46.61,  0,
446,    46.14,  0,
447,    45.86,  0,
448,    45.89,  0,
449,    45.76,  0,
450,    45.18,  0,
451,    44.31,  0,
452,    43.27,  0,
453,    41.85,  0,
454,    39.69,  0,
455,    36.81,  0,
456,    33.66,  0,
457,    30.55,  0,
458,    27.25,  0,
459,    23.77,  0,
460,    20.85,  0,
461,    18.65,  0,
462,    16.41,  0,
463,    13.89,  0,
464,    11.8,   0,
465,    10.42,  0,
466,    9.38,   0,
467,    8.61,   0,
468,    8.14,   0,
469,    7.47,   0,
470,    6.43,   0,
471,    4.35,   0,
472,    2.49,   0,
473,    1.27,   0,
474,    0,  0,
475,    0,  0,
476,    0,  0,
477,    0,  0,
478,    0,  0,
479,    0,  0,
480,    0,  0,
481,    0,  0,
482,    0,  0,
483,    0,  0,
484,    0,  0,
485,    0,  0,
486,    0,  0,
487,    0,  0,
488,    0,  0,
489,    0,  0,
490,    0,  0,
491,    0,  0,
492,    0,  0,
493,    0,  0,
494,    0,  0,
495,    0,  0,
496,    0,  0,
497,    0,  0,
498,    0,  0,
499,    0,  0,
500,    0,  0,
501,    0,  0,
502,    0,  0,
503,    0,  0,
504,    0,  0,
505,    0,  0,
506,    0,  0,
507,    0,  0,
508,    0,  0,
509,    0,  0,
510,    0,  0,
511,    0,  0,
512,    0,  0,
513,    0,  0,
514,    0,  0,
515,    0,  0,
516,    0,  0,
517,    0,  0,
518,    0,  0,
519,    0,  0,
520,    0,  0,
521,    3.37,   0,
522,    9.1,    0,
523,    14.02,  0,
524,    17.2,   0,
525,    20.22,  0,
```

```
526,    23.49,  0,
527,    26.43,  0,
528,    28.9,   0,
529,    30.55,  0,
530,    31.17,  0,
531,    31.42,  0,
532,    31.48,  0,
533,    30.84,  0,
534,    29.9,   0,
535,    29.66,  0,
536,    29.2,   0,
537,    28.45,  0,
538,    27.4,   0,
539,    26.21,  0,
540,    25.27,  0,
541,    24.81,  0,
542,    24.97,  0,
543,    26.03,  0,
544,    27.81,  0,
545,    29.48,  0,
546,    30.48,  0,
547,    30.85,  0,
548,    30.59,  0,
549,    29.84,  0,
550,    28.92,  0,
551,    27.47,  0,
552,    24.78,  0,
553,    21.41,  0,
554,    18.66,  0,
555,    16.85,  0,
556,    15.79,  0,
557,    16.08,  0,
558,    18.06,  0,
559,    21.01,  0,
560,    24.26,  0,
561,    27.72,  0,
562,    31.07,  0,
563,    33.82,  0,
564,    35.9,   0,
565,    37.26,  0,
566,    37.71,  0,
567,    37.5,   0,
568,    37.07,  0,
569,    36.47,  0,
570,    35.57,  0,
571,    34.41,  0,
572,    33.12,  0,
573,    31.87,  0,
574,    30.79,  0,
575,    29.85,  0,
576,    28.93,  0,
577,    28.08,  0,
578,    27.6,   0,
579,    28.02,  0,
580,    29.68,  0,
581,    31.96,  0,
582,    33.94,  0,
583,    35.57,  0,
584,    37.21,  0,
585,    38.51,  0,
586,    39.39,  0,
587,    40.58,  0,
588,    42.2,   0,
589,    43.44,  0,
590,    44.19,  0,
591,    44.96,  0,
592,    45.73,  0,
593,    46.29,  0,
594,    46.87,  0,
595,    47.51,  0,
596,    48.07,  0,
597,    48.82,  0,
598,    49.85,  0,
599,    50.68,  0,
600,    51.26,  0,
601,    52.04,  0,
602,    52.82,  0,
603,    53.22,  0,
604,    53.53,  0,
605,    54, 0,
606,    54.31,  0,
607,    54.35,  0,
608,    54.37,  0,
609,    54.28,  0,
610,    53.91,  0,
611,    53.18,  0,
612,    51.82,  0,
613,    49.83,  0,
```

```
614,    47.71,  0,
615,    45.39,  0,
616,    41.8,   0,
617,    37.47,  0,
618,    33.19,  0,
619,    30.27,  0,
620,    26.16,  0,
621,    19.57,  0,
622,    13.81,  0,
623,    11.04,  0,
624,    9.11,   0,
625,    6.17,   0,
626,    3.13,   0,
627,    1.17,   0,
628,    0,  0,
629,    0,  0,
630,    0,  0,
631,    0,  0,
632,    0,  0,
633,    0,  0,
634,    0,  0,
635,    0,  0,
636,    0,  0,
637,    0,  0,
638,    0,  0,
639,    0,  0,
640,    0,  0,
641,    0,  0,
642,    0,  0,
643,    0,  0,
644,    0,  0,
645,    0,  0,
646,    0,  0,
647,    0,  0,
648,    0,  0,
649,    0,  0,
650,    0,  0,
651,    0,  0,
652,    0,  0,
653,    0,  0,
654,    0,  0,
655,    0,  0,
656,    0,  0,
657,    0,  0,
658,    0,  0,
659,    0,  0,
660,    3.83,   0,
661,    9.38,   0,
662,    13.85,  0,
663,    14.91,  0,
664,    15.68,  0,
665,    19.52,  0,
666,    24.58,  0,
667,    27.2,   0,
668,    27.48,  0,
669,    27.85,  0,
670,    29.15,  0,
671,    31.13,  0,
672,    33.52,  0,
673,    35.89,  0,
674,    37.09,  0,
675,    37.33,  0,
676,    37.1,   0,
677,    36.3,   0,
678,    35.03,  0,
679,    34.21,  0,
680,    34.23,  0,
681,    34.31,  0,
682,    33.99,  0,
683,    33.82,  0,
684,    34.34,  0,
685,    35.49,  0,
686,    37.22,  0,
687,    39.53,  0,
688,    41.98,  0,
689,    44.08,  0,
690,    45.69,  0,
691,    46.78,  0,
692,    47.45,  0,
693,    47.84,  0,
694,    47.82,  0,
695,    47.14,  0,
696,    46.06,  0,
697,    45.13,  0,
698,    44.55,  0,
699,    44.41,  0,
700,    44.84,  0,
701,    45.56,  0,
```

```
702,    45.84,  0,
703,    45.28,  0,
704,    43.79,  0,
705,    41.57,  0,
706,    39, 0,
707,    36.35,  0,
708,    33.6,   0,
709,    30.97,  0,
710,    28.86,  0,
711,    27, 0,
712,    24.95,  0,
713,    23.05,  0,
714,    21.71,  0,
715,    20.52,  0,
716,    19.39,  0,
717,    19.06,  0,
718,    19.7,   0,
719,    20.5,   0,
720,    20.95,  0,
721,    21.18,  0,
722,    21.19,  0,
723,    20.66,  0,
724,    19.26,  0,
725,    16.67,  0,
726,    13.34,  0,
727,    10.48,  0,
728,    8.59,   0,
729,    6.93,   0,
730,    4.36,   0,
731,    2.09,   0,
732,    0,  0,
733,    0,  0,
734,    0,  0,
735,    0,  0,
736,    0,  0,
737,    0,  0,
738,    0,  0,
739,    0,  0,
740,    0,  0,
741,    0,  0,
742,    0,  0,
743,    0,  0,
744,    0,  0,
745,    0,  0,
746,    0,  0,
747,    0,  0,
748,    0,  0,
749,    0,  0,
750,    1.05,   0,
751,    5.67,   0,
752,    9.44,   0,
753,    13.24,  0,
754,    16.38,  0,
755,    18.36,  0,
756,    19.93,  0,
757,    22.25,  0,
758,    25.25,  0,
759,    28.34,  0,
760,    31.32,  0,
761,    33.95,  0,
762,    35.96,  0,
763,    37.89,  0,
764,    40.21,  0,
765,    42.12,  0,
766,    42.93,  0,
767,    43.53,  0,
768,    44.8,   0,
769,    46.02,  0,
770,    46.29,  0,
771,    46.15,  0,
772,    46.42,  0,
773,    47.03,  0,
774,    47.57,  0,
775,    48.1,   0,
776,    48.68,  0,
777,    49.16,  0,
778,    49.56,  0,
779,    50.16,  0,
780,    50.97,  0,
781,    51.75,  0,
782,    52.42,  0,
783,    53, 0,
784,    53.38,  0,
785,    53.57,  0,
786,    53.7,   0,
787,    53.61,  0,
788,    53.06,  0,
789,    52.29,  0,
```

```
790,    51.78,  0,
791,    51.48,  0,
792,    50.93,  0,
793,    49.93,  0,
794,    48.45,  0,
795,    46.42,  0,
796,    43.97,  0,
797,    41.48,  0,
798,    39.39,  0,
799,    38.18,  0,
800,    38.09,  0,
801,    38.7,   0,
802,    39.19,  0,
803,    39.06,  0,
804,    38.27,  0,
805,    37.02,  0,
806,    35.67,  0,
807,    34.61,  0,
808,    33.89,  0,
809,    33.32,  0,
810,    32.62,  0,
811,    31.41,  0,
812,    29.63,  0,
813,    27.83,  0,
814,    26.44,  0,
815,    25.4,   0,
816,    24.84,  0,
817,    25.24,  0,
818,    26.34,  0,
819,    27.09,  0,
820,    27.12,  0,
821,    27.01,  0,
822,    27.21,  0,
823,    27.7,   0,
824,    28.48,  0,
825,    29.54,  0,
826,    30.6,   0,
827,    31.61,  0,
828,    32.8,   0,
829,    34.11,  0,
830,    35.2,   0,
831,    36.1,   0,
832,    37.13,  0,
833,    38.13,  0,
834,    38.62,  0,
835,    38.6,   0,
836,    38.48,  0,
837,    38.23,  0,
838,    37.4,   0,
839,    35.99,  0,
840,    34.45,  0,
841,    33.07,  0,
842,    31.81,  0,
843,    30.59,  0,
844,    29.4,   0,
845,    28.4,   0,
846,    27.63,  0,
847,    26.57,  0,
848,    24.25,  0,
849,    20.69,  0,
850,    14.6,   0,
851,    8.99,   0,
852,    4.76,   0,
853,    1.64,   0,
854,    0,  0,
855,    0,  0,
856,    0,  0,
857,    0,  0,
858,    0,  0,
859,    0,  0,
860,    0,  0,
861,    0,  0,
862,    0,  0,
863,    0,  0,
864,    0,  0,
865,    0,  0,
866,    0,  0,
867,    0,  0,
868,    0,  0,
869,    0,  0,
870,    0,  0,
871,    0,  0,
872,    0,  0,
873,    0,  0,
874,    0,  0,
875,    0,  0,
876,    0,  0,
877,    0,  0,
```

```
878,    0,  0,
879,    0,  0,
880,    0,  0,
881,    0,  0,
882,    0,  0,
883,    0,  0,
884,    0,  0,
885,    0,  0,
886,    0,  0,
887,    0,  0,
888,    0,  0,
889,    0,  0,
890,    0,  0,
891,    3.57,   0,
892,    8.28,   0,
893,    11.75,  0,
894,    13.06,  0,
895,    15.07,  0,
896,    18.64,  0,
897,    21.16,  0,
898,    22.19,  0,
899,    22.89,  0,
900,    23.73,  0,
901,    23.37,  0,
902,    22.87,  0,
903,    22.73,  0,
904,    22.51,  0,
905,    22.01,  0,
906,    21.45,  0,
907,    21.23,  0,
908,    22.02,  0,
909,    23.88,  0,
910,    25.74,  0,
911,    26.82,  0,
912,    27.78,  0,
913,    29.33,  0,
914,    31.26,  0,
915,    33.32,  0,
916,    35.53,  0,
917,    37.6,   0,
918,    39.26,  0,
919,    40.64,  0,
920,    41.7,   0,
921,    42.23,  0,
922,    42.5,   0,
923,    42.75,  0,
924,    42.61,  0,
925,    41.89,  0,
926,    40.86,  0,
927,    39.56,  0,
928,    37.87,  0,
929,    36.03,  0,
930,    34.13,  0,
931,    31.63,  0,
932,    27.79,  0,
933,    22.97,  0,
934,    18.01,  0,
935,    13.36,  0,
936,    9.31,   0,
937,    6.7,    0,
938,    5.31,   0,
939,    3.98,   0,
940,    2.54,   0,
941,    1.4,    0,
942,    0,  0,
943,    0,  0,
944,    0,  0,
945,    0,  0,
946,    0,  0,
947,    0,  0,
948,    0,  0,
949,    0,  0,
950,    0,  0,
951,    0,  0,
952,    0,  0,
953,    0,  0,
954,    0,  0,
955,    0,  0,
956,    0,  0,
957,    0,  0,
958,    0,  0,
959,    0,  0,
960,    0,  0,
961,    0,  0,
962,    0,  0,
963,    0,  0,
964,    0,  0,
965,    0,  0,
```

```
966,    0,  0,
967,    0,  0,
968,    0,  0,
969,    0,  0,
970,    0,  0,
971,    0,  0,
972,    0,  0,
973,    0,  0,
974,    0,  0,
975,    0,  0,
976,    0,  0,
977,    0,  0,
978,    0,  0,
979,    0,  0,
980,    0,  0,
981,    0,  0,
982,    0,  0,
983,    0,  0,
984,    0,  0,
985,    0,  0,
986,    0,  0,
987,    0,  0,
988,    0,  0,
989,    0,  0,
990,    0,  0,
991,    0,  0,
992,    0,  0,
993,    0,  0,
994,    0,  0,
995,    0,  0,
996,    2.62,   0,
997,    3.82,   0,
998,    4.08,   0,
999,    6.12,   0,
1000,   8.81,   0,
1001,   9.73,   0,
1002,   9.59,   0,
1003,   9.44,   0,
1004,   9.45,   0,
1005,   9.35,   0,
1006,   9.3,    0,
1007,   9.75,   0,
1008,   10.7,   0,
1009,   11.61,  0,
1010,   12.02,  0,
1011,   12.02,  0,
1012,   11.71,  0,
1013,   10.78,  0,
1014,   9.34,   0,
1015,   6.66,   0,
1016,   4.63,   0,
1017,   3.28,   0,
1018,   1.7,    0,
1019,   0,  0,
1020,   0,  0,
1021,   0,  0,
1022,   0,  0,
1023,   0,  0,
1024,   2.43,   0,
1025,   4.63,   0,
1026,   7.93,   0,
1027,   9.13,   0,
1028,   10.21,  0,
1029,   11.28,  0,
1030,   12.87,  0,
1031,   14.44,  0,
1032,   15.28,  0,
1033,   15.41,  0,
1034,   15.33,  0,
1035,   15.28,  0,
1036,   14.97,  0,
1037,   14.23,  0,
1038,   13.7,   0,
1039,   14.26,  0,
1040,   15.77,  0,
1041,   17.25,  0,
1042,   18.21,  0,
1043,   18.82,  0,
1044,   19, 0,
1045,   18.44,  0,
1046,   17.29,  0,
1047,   16.12,  0,
1048,   15, 0,
1049,   13.52,  0,
1050,   11.83,  0,
1051,   10.76,  0,
1052,   10.49,  0,
1053,   10.04,  0,
```

```
1054,   8.94,   0,
1055,   8.11,   0,
1056,   8.15,   0,
1057,   8.24,   0,
1058,   7.77,   0,
1059,   7.65,   0,
1060,   8.64,   0,
1061,   10.04,  0,
1062,   10.94,  0,
1063,   11.29,  0,
1064,   11.36,  0,
1065,   11.01,  0,
1066,   10.01,  0,
1067,   8.54,   0,
1068,   7.13,   0,
1069,   6.41,   0,
1070,   6.79,   0,
1071,   8.38,   0,
1072,   10.73,  0,
1073,   12.83,  0,
1074,   14.04,  0,
1075,   14.97,  0,
1076,   16.4,   0,
1077,   18.03,  0,
1078,   19.52,  0,
1079,   21.53,  0,
1080,   24.25,  0,
1081,   26.42,  0,
1082,   27.3,   0,
1083,   27.75,  0,
1084,   28.38,  0,
1085,   28.62,  0,
1086,   28.01,  0,
1087,   26.91,  0,
1088,   25.46,  0,
1089,   23.49,  0,
1090,   20.45,  0,
1091,   17.47,  0,
1092,   14.8,   0,
1093,   12.03,  0,
1094,   9.34,   0,
1095,   7.27,   0,
1096,   5.43,   0,
1097,   3.23,   0,
1098,   1.22,   0,
1099,   0,  0,
1100,   0,  0,
1101,   0,  0,
1102,   0,  0,
1103,   0,  0,
1104,   0,  0,
1105,   0,  0,
1106,   0,  0,
1107,   0,  0,
1108,   0,  0,
1109,   0,  0,
1110,   0,  0,
1111,   0,  0,
1112,   0,  0,
1113,   0,  0,
1114,   0,  0,
1115,   0,  0,
1116,   0,  0,
1117,   0,  0,
1118,   0,  0,
1119,   0,  0,
1120,   0,  0,
1121,   0,  0,
1122,   0,  0,
1123,   0,  0,
1124,   0,  0,
1125,   0,  0,
1126,   0,  0,
1127,   0,  0,
1128,   0,  0,
1129,   0,  0,
1130,   0,  0,
1131,   0,  0,
1132,   0,  0,
1133,   0,  0,
1134,   0,  0,
1135,   0,  0,
1136,   0,  0,
1137,   0,  0,
1138,   0,  0,
1139,   0,  0,
1140,   0,  0,
1141,   1.92,   0,
```

```
1142,   3.93,   0,
1143,   6.8,    0,
1144,   9.57,   0,
1145,   12.26,  0,
1146,   13.88,  0,
1147,   14.61,  0,
1148,   15.12,  0,
1149,   15.52,  0,
1150,   15.14,  0,
1151,   13.51,  0,
1152,   11.06,  0,
1153,   8.82,   0,
1154,   7.51,   0,
1155,   7.24,   0,
1156,   7.54,   0,
1157,   7.69,   0,
1158,   7.12,   0,
1159,   5.85,   0,
1160,   3.9,    0,
1161,   2.23,   0,
1162,   1.49,   0,
1163,   0,  0,
1164,   0,  0,
1165,   0,  0,
1166,   0,  0,
1167,   0,  0,
1168,   0,  0,
1169,   0,  0,
1170,   0,  0,
1171,   1.08,   0,
1172,   1.34,   0,
1173,   3.04,   0,
1174,   3.84,   0,
1175,   4.07,   0,
1176,   5.12,   0,
1177,   7.12,   0,
1178,   9.07,   0,
1179,   10.25,  0,
1180,   10.65,  0,
1181,   10.61,  0,
1182,   10.78,  0,
1183,   11.61,  0,
1184,   12.65,  0,
1185,   13.2,   0,
1186,   13.16,  0,
1187,   12.95,  0,
1188,   12.77,  0,
1189,   12.5,   0,
1190,   12.07,  0,
1191,   11.66,  0,
1192,   11.35,  0,
1193,   10.77,  0,
1194,   9.56,   0,
1195,   8.03,   0,
1196,   6.72,   0,
1197,   5.73,   0,
1198,   4.94,   0,
1199,   4.46,   0,
1200,   4.29,   0,
1201,   4.15,   0,
1202,   3.85,   0,
1203,   3.31,   0,
1204,   2.49,   0,
1205,   1.33,   0,
1206,   0,  0,
1207,   0,  0,
1208,   0,  0,
1209,   0,  0,
1210,   0,  0,
1211,   0,  0,
1212,   0,  0,
1213,   0,  0,
1214,   0,  0,
1215,   0,  0,
1216,   0,  0,
1217,   0,  0,
1218,   0,  0,
1219,   0,  0,
1220,   0,  0,
1221,   0,  0,
1222,   0,  0,
1223,   0,  0,
1224,   0,  0,
1225,   0,  0,
1226,   0,  0,
1227,   0,  0,
1228,   0,  0,
1229,   0,  0,
```

```
1230,   0,  0,
1231,   0,  0,
1232,   0,  0,
1233,   0,  0,
1234,   0,  0,
1235,   0,  0,
1236,   0,  0,
1237,   0,  0,
1238,   0,  0,
1239,   0,  0,
1240,   0,  0,
1241,   0,  0,
1242,   0,  0,
1243,   0,  0,
1244,   0,  0,
1245,   0,  0,
1246,   0,  0,
1247,   0,  0,
1248,   0,  0,
1249,   0,  0,
1250,   0,  0,
1251,   0,  0,
1252,   0,  0,
1253,   0,  0,
1254,   0,  0,
1255,   0,  0,
1256,   0,  0,
1257,   0,  0,
1258,   0,  0,
1259,   0,  0,
1260,   0,  0,
1261,   0,  0,
1262,   0,  0,
1263,   0,  0,
1264,   0,  0,
1265,   0,  0,
1266,   0,  0,
1267,   0,  0,
1268,   0,  0,
1269,   0,  0,
1270,   0,  0,
1271,   0,  0,
1272,   0,  0,
1273,   0,  0,
1274,   0,  0,
1275,   0,  0,
1276,   0,  0,
1277,   0,  0,
1278,   0,  0,
1279,   0,  0,
1280,   0,  0,
1281,   0,  0,
1282,   0,  0,
1283,   0,  0,
1284,   0,  0,
1285,   0,  0,
1286,   0,  0,
1287,   0,  0,
1288,   0,  0,
1289,   0,  0,
1290,   1.28,   0,
1291,   1.6,    0,
1292,   2.63,   0,
1293,   5.02,   0,
1294,   8.68,   0,
1295,   12.57,  0,
1296,   15.07,  0,
1297,   16.22,  0,
1298,   17.46,  0,
1299,   19.65,  0,
1300,   20.82,  0,
1301,   21.47,  0,
1302,   22.09,  0,
1303,   22.09,  0,
1304,   20.95,  0,
1305,   18.99,  0,
1306,   16.56,  0,
1307,   14.08,  0,
1308,   12.39,  0,
1309,   11.84,  0,
1310,   11.86,  0,
1311,   12.11,  0,
1312,   13.01,  0,
1313,   14.67,  0,
1314,   16.56,  0,
1315,   18.29,  0,
1316,   20.07,  0,
1317,   22.45,  0,
```

```
1318,   25.37,  0,
1319,   27.84,  0,
1320,   29.36,  0,
1321,   30.76,  0,
1322,   32.49,  0,
1323,   33.61,  0,
1324,   33.67,  0,
1325,   33.55,  0,
1326,   33.29,  0,
1327,   32.04,  0,
1328,   30.09,  0,
1329,   28.23,  0,
1330,   26.18,  0,
1331,   23.77,  0,
1332,   22.06,  0,
1333,   21.48,  0,
1334,   21.25,  0,
1335,   21.09,  0,
1336,   21.08,  0,
1337,   20.47,  0,
1338,   18.82,  0,
1339,   16.86,  0,
1340,   14.85,  0,
1341,   11.76,  0,
1342,   8.45,   0,
1343,   5.33,   0,
1344,   3.78,   0,
1345,   2.45,   0,
1346,   0,  0,
1347,   0,  0,
1348,   0,  0,
1349,   0,  0,
1350,   0,  0,
1351,   0,  0,
1352,   0,  0,
1353,   0,  0,
1354,   0,  0,
1355,   0,  0,
1356,   0,  0,
1357,   0,  0,
1358,   0,  0,
1359,   1.86,   0,
1360,   6.31,   0,
1361,   9.9,    0,
1362,   12.02,  0,
1363,   13.52,  0,
1364,   15.04,  0,
1365,   14.83,  0,
1366,   13.43,  0,
1367,   12.27,  0,
1368,   12.79,  0,
1369,   14.79,  0,
1370,   16.84,  0,
1371,   18.64,  0,
1372,   20.87,  0,
1373,   23.02,  0,
1374,   24.13,  0,
1375,   24.6,   0,
1376,   24.92,  0,
1377,   24.67,  0,
1378,   23.86,  0,
1379,   22.97,  0,
1380,   21.5,   0,
1381,   19.1,   0,
1382,   16.7,   0,
1383,   15.04,  0,
1384,   13.91,  0,
1385,   13.35,  0,
1386,   13.4,   0,
1387,   13.35,  0,
1388,   12.77,  0,
1389,   11.82,  0,
1390,   9.99,   0,
1391,   7.19,   0,
1392,   5.07,   0,
1393,   4.85,   0,
1394,   5.29,   0,
1395,   4.82,   0,
1396,   3.66,   0,
1397,   1.87,   0,
1398,   0,  0,
1399,   0,  0,
1400,   0,  0,
1401,   0,  0,
1402,   0,  0,
1403,   0,  0,
1404,   0,  0,
1405,   0,  0,
```

```
1406,   0,  0,
1407,   0,  0,
1408,   0,  0,
1409,   0,  0,
1410,   0,  0,
1411,   0,  0,
1412,   0,  0,
1413,   0,  0,
1414,   0,  0,
1415,   0,  0,
1416,   0,  0,
1417,   0,  0,
1418,   0,  0,
1419,   0,  0,
1420,   0,  0,
1421,   0,  0,
1422,   0,  0,
1423,   0,  0,
1424,   0,  0,
1425,   3.5,    0,
1426,   5.08,   0,
1427,   5.97,   0,
1428,   9.46,   0,
1429,   13.96,  0,
1430,   15.88,  0,
1431,   16.84,  0,
1432,   19.06,  0,
1433,   21.53,  0,
1434,   23.63,  0,
1435,   25.88,  0,
1436,   28.25,  0,
1437,   30.55,  0,
1438,   32.83,  0,
1439,   34.81,  0,
1440,   36.22,  0,
1441,   37.19,  0,
1442,   38.01,  0,
1443,   38.69,  0,
1444,   39.31,  0,
1445,   40.16,  0,
1446,   41.24,  0,
1447,   42.33,  0,
1448,   43.38,  0,
1449,   44.56,  0,
1450,   45.85,  0,
1451,   47.02,  0,
1452,   47.93,  0,
1453,   48.8,   0,
1454,   49.73,  0,
1455,   50.57,  0,
1456,   51.32,  0,
1457,   52.19,  0,
1458,   53.16,  0,
1459,   53.98,  0,
1460,   54.72,  0,
1461,   55.55,  0,
1462,   56.47,  0,
1463,   57.48,  0,
1464,   58.69,  0,
1465,   60, 0,
1466,   61.2,   0,
1467,   62.42,  0,
1468,   63.75,  0,
1469,   65.05,  0,
1470,   66.16,  0,
1471,   67.12,  0,
1472,   67.89,  0,
1473,   68.54,  0,
1474,   69.22,  0,
1475,   69.98,  0,
1476,   70.71,  0,
1477,   71.47,  0,
1478,   72.36,  0,
1479,   73.35,  0,
1480,   74.41,  0,
1481,   75.52,  0,
1482,   76.52,  0,
1483,   77.39,  0,
1484,   78.29,  0,
1485,   79.22,  0,
1486,   79.95,  0,
1487,   80.45,  0,
1488,   80.88,  0,
1489,   81.25,  0,
1490,   81.56,  0,
1491,   81.81,  0,
1492,   81.86,  0,
1493,   81.66,  0,
```

```
1494,   81.19,  0,
1495,   80.68,  0,
1496,   80.44,  0,
1497,   80.39,  0,
1498,   80.29,  0,
1499,   80.21,  0,
1500,   80.19,  0,
1501,   80.03,  0,
1502,   79.63,  0,
1503,   79.25,  0,
1504,   79.09,  0,
1505,   79.08,  0,
1506,   79.01,  0,
1507,   78.84,  0,
1508,   78.61,  0,
1509,   78.44,  0,
1510,   78.34,  0,
1511,   78.23,  0,
1512,   78.15,  0,
1513,   78.19,  0,
1514,   78.28,  0,
1515,   78.34,  0,
1516,   78.46,  0,
1517,   78.72,  0,
1518,   79.03,  0,
1519,   79.3,   0,
1520,   79.61,  0,
1521,   79.99,  0,
1522,   80.39,  0,
1523,   80.75,  0,
1524,   81.08,  0,
1525,   81.39,  0,
1526,   81.73,  0,
1527,   82.05,  0,
1528,   82.37,  0,
1529,   82.74,  0,
1530,   83.1,   0,
1531,   83.34,  0,
1532,   83.46,  0,
1533,   83.51,  0,
1534,   83.42,  0,
1535,   83.22,  0,
1536,   83.08,  0,
1537,   82.97,  0,
1538,   82.72,  0,
1539,   82.41,  0,
1540,   82.17,  0,
1541,   81.84,  0,
1542,   81.34,  0,
1543,   80.89,  0,
1544,   80.63,  0,
1545,   80.42,  0,
1546,   80.17,  0,
1547,   79.93,  0,
1548,   79.67,  0,
1549,   79.45,  0,
1550,   79.42,  0,
1551,   79.5,   0,
1552,   79.5,   0,
1553,   79.53,  0,
1554,   79.72,  0,
1555,   79.88,  0,
1556,   79.81,  0,
1557,   79.69,  0,
1558,   79.75,  0,
1559,   79.95,  0,
1560,   80.24,  0,
1561,   80.68,  0,
1562,   81.25,  0,
1563,   81.84,  0,
1564,   82.39,  0,
1565,   82.9,   0,
1566,   83.42,  0,
1567,   83.92,  0,
1568,   84.34,  0,
1569,   84.67,  0,
1570,   84.94,  0,
1571,   85.12,  0,
1572,   85.09,  0,
1573,   84.86,  0,
1574,   84.51,  0,
1575,   84.09,  0,
1576,   83.66,  0,
1577,   83.3,   0,
1578,   82.94,  0,
1579,   82.54,  0,
1580,   82.18,  0,
1581,   81.96,  0,
```

```
1582,   81.86,  0,
1583,   81.85,  0,
1584,   81.82,  0,
1585,   81.64,  0,
1586,   81.37,  0,
1587,   81.15,  0,
1588,   80.89,  0,
1589,   80.5,   0,
1590,   80.25,  0,
1591,   80.39,  0,
1592,   80.83,  0,
1593,   81.44,  0,
1594,   82.31,  0,
1595,   83.38,  0,
1596,   84.39,  0,
1597,   85.24,  0,
1598,   86, 0,
1599,   86.67,  0,
1600,   87.2,   0,
1601,   87.55,  0,
1602,   87.6,   0,
1603,   87.39,  0,
1604,   87.1,   0,
1605,   86.87,  0,
1606,   86.62,  0,
1607,   86.35,  0,
1608,   86.17,  0,
1609,   85.99,  0,
1610,   85.77,  0,
1611,   85.59,  0,
1612,   85.51,  0,
1613,   85.45,  0,
1614,   85.43,  0,
1615,   85.61,  0,
1616,   85.99,  0,
1617,   86.3,   0,
1618,   86.45,  0,
1619,   86.5,   0,
1620,   86.57,  0,
1621,   86.66,  0,
1622,   86.79,  0,
1623,   86.98,  0,
1624,   87.08,  0,
1625,   86.85,  0,
1626,   86.16,  0,
1627,   85.28,  0,
1628,   84.52,  0,
1629,   83.98,  0,
1630,   83.51,  0,
1631,   83.1,   0,
1632,   82.77,  0,
1633,   82.6,   0,
1634,   81.91,  0,
1635,   80.94,  0,
1636,   79.82,  0,
1637,   78.5,   0,
1638,   77, 0,
1639,   75.57,  0,
1640,   74.34,  0,
1641,   73.14,  0,
1642,   71.88,  0,
1643,   70.73,  0,
1644,   69.59,  0,
1645,   67.81,  0,
1646,   64.91,  0,
1647,   60.93,  0,
1648,   56.12,  0,
1649,   50.87,  0,
1650,   45.7,   0,
1651,   40.78,  0,
1652,   35.82,  0,
1653,   30.85,  0,
1654,   26.48,  0,
1655,   23.12,  0,
1656,   20.59,  0,
1657,   18.47,  0,
1658,   16.69,  0,
1659,   15.82,  0,
1660,   15.57,  0,
1661,   15.98,  0,
1662,   17.14,  0,
1663,   18.68,  0,
1664,   20.11,  0,
1665,   21.3,   0,
1666,   22.22,  0,
1667,   22.5,   0,
1668,   22.13,  0,
1669,   21.85,  0,
```

```
1670,   22.02,  0,
1671,   22.17,  0,
1672,   21.86,  0,
1673,   21.08,  0,
1674,   19.5,   0,
1675,   16.78,  0,
1676,   13.55,  0,
1677,   11.03,  0,
1678,   9.72,   0,
1679,   9.38,   0,
1680,   9.55,   0,
1681,   9.71,   0,
1682,   9.65,   0,
1683,   9.8,    0,
1684,   10.85,  0,
1685,   12.8,   0,
1686,   15.13,  0,
1687,   17.67,  0,
1688,   20.63,  0,
1689,   23.74,  0,
1690,   26.17,  0,
1691,   27.49,  0,
1692,   28.09,  0,
1693,   28.36,  0,
1694,   28.16,  0,
1695,   27.31,  0,
1696,   26.07,  0,
1697,   24.71,  0,
1698,   23.29,  0,
1699,   22, 0,
1700,   20.98,  0,
1701,   19.93,  0,
1702,   18.57,  0,
1703,   17.29,  0,
1704,   16.67,  0,
1705,   16.69,  0,
1706,   17.09,  0,
1707,   17.92,  0,
1708,   19.14,  0,
1709,   20.34,  0,
1710,   21.1,   0,
1711,   21.3,   0,
1712,   21.06,  0,
1713,   20.63,  0,
1714,   20.33,  0,
1715,   20.44,  0,
1716,   20.97,  0,
1717,   21.6,   0,
1718,   21.76,  0,
1719,   21.39,  0,
1720,   21.23,  0,
1721,   21.63,  0,
1722,   21.9,   0,
1723,   21.45,  0,
1724,   20.74,  0,
1725,   20.26,  0,
1726,   19.76,  0,
1727,   19.11,  0,
1728,   18.79,  0,
1729,   18.97,  0,
1730,   19.31,  0,
1731,   19.9,   0,
1732,   21.06,  0,
1733,   22.54,  0,
1734,   23.8,   0,
1735,   24.79,  0,
1736,   25.59,  0,
1737,   26.01,  0,
1738,   25.83,  0,
1739,   25.26,  0,
1740,   24.73,  0,
1741,   24.39,  0,
1742,   23.94,  0,
1743,   23.3,   0,
1744,   23.1,   0,
1745,   23.72,  0,
1746,   24.49,  0,
1747,   24.77,  0,
1748,   25.01,  0,
1749,   25.58,  0,
1750,   25.92,  0,
1751,   25.88,  0,
1752,   26.08,  0,
1753,   26.44,  0,
1754,   26.17,  0,
1755,   25.39,  0,
1756,   24.87,  0,
1757,   24.61,  0,
```

```
1758,   24.22,  0,
1759,   23.93,  0,
1760,   24.01,  0,
1761,   24, 0,
1762,   23.27,  0,
1763,   22.03,  0,
1764,   21.23,  0,
1765,   21.51,  0,
1766,   22.53,  0,
1767,   23.61,  0,
1768,   24.63,  0,
1769,   25.66,  0,
1770,   26.14,  0,
1771,   25.76,  0,
1772,   25.08,  0,
1773,   24.34,  0,
1774,   22.99,  0,
1775,   21.14,  0,
1776,   19.79,  0,
1777,   19.14,  0,
1778,   18.49,  0,
1779,   17.6,   0,
1780,   16.83,  0,
1781,   16.34,  0,
1782,   16.15,  0,
1783,   16.24,  0,
1784,   16.37,  0,
1785,   16.26,  0,
1786,   15.85,  0,
1787,   15.12,  0,
1788,   14.32,  0,
1789,   13.93,  0,
1790,   13.94,  0,
1791,   13.75,  0,
1792,   13.41,  0,
1793,   13.58,  0,
1794,   14.32,  0,
1795,   15.23,  0,
1796,   16.18,  0,
1797,   16.91,  0,
1798,   16.85,  0,
1799,   16.2,   0,
1800,   15.78,  0,
1801,   15.84,  0,
1802,   16.19,  0,
1803,   16.95,  0,
1804,   17.97,  0,
1805,   18.49,  0,
1806,   18.03,  0,
1807,   16.97,  0,
1808,   16.16,  0,
1809,   16.41,  0,
1810,   17.91,  0,
1811,   19.7,   0,
1812,   20.54,  0,
1813,   20.4,   0,
1814,   20.22,  0,
1815,   20.55,  0,
1816,   21.16,  0,
1817,   21.53,  0,
1818,   21.28,  0,
1819,   20.29,  0,
1820,   18.95,  0,
1821,   17.79,  0,
1822,   16.89,  0,
1823,   15.98,  0,
1824,   15.16,  0,
1825,   13.24,  0,
1826,   10.27,  0,
1827,   5.06,   0,
1828,   0,  0,
1829,   0,  0};
```

```
#define MODE2_SIZE  3120
double mode2[MODE2_SIZE][3] = {
0,  80, 0,
1,  80, 0,
2,  80, 0,
3,  80, 0,
4,  80, -0.228,
5,  80, -2.28,
6,  80, -2.28,
7,  80, -2.28,
8,  80, -2.28,
9,  80, -2.28,
10, 80, -2.28,
11, 80, -2.28,
12, 80, -0.669,
13, 80, -0.49,
14, 80, -0.49,
15, 80, -0.49,
16, 80, -0.49,
17, 80, -0.49,
18, 80, 0.3,
19, 80, 0.494,
20, 80, 1.27,
21, 80, 1.27,
22, 80, 3.135,
23, 80, 5,
24, 80, 5,
25, 80, 1.4,
26, 80, -4,
27, 80, -4,
28, 80, -4,
29, 80, -1.78,
30, 80, 0.712,
31, 80, -1.08,
32, 80, -1.08,
33, 80, -1.08,
34, 80, 2.2,
35, 80, 2.2,
36, 80, 2.2,
37, 80, 2.2,
38, 80, 2.2,
39, 80, 2.2,
40, 80, 2.2,
41, 80, 2.2,
42, 80, 2.2,
43, 80, 1.162,
44, 80, -2.99,
45, 80, -2.99,
46, 80, -2.99,
47, 80, -2.99,
48, 80, -2.99,
49, 80, -2.639,
50, 80, 0.52,
51, 80, 0.52,
52, 80, 0.52,
53, 80, 0.52,
54, 80, 0.52,
55, 80, 0.768,
56, 80, 0.83,
57, 80, 0.83,
58, 80, 0.83,
59, 80, 0.83,
60, 80, -2.91,
61, 80, -2.91,
62, 80, -2.91,
63, 80, -2.91,
64, 80, -2.91,
65, 80, 1.786,
66, 80, 2.96,
67, 80, 2.96,
68, 80, 2.96,
69, 80, 2.96,
70, 80, 2.96,
71, 80, 0.576,
72, 80, -3,
73, 80, -3,
74, 80, -3,
75, 80, -3,
76, 80, -3,
77, 80, -3,
78, 80, 2.247,
79, 80, 2.83,
80, 80, 2.83,
81, 80, 2.83,
82, 80, 2.83,
83, 80, 2.83,
84, 80, 2.83,
85, 80, 2.83,
```

```
86, 80, 2.83,
87, 80, 1.122,
88, 80, 0.996,
89, 80, 2.41,
90, 80, 2.41,
91, 80, 2.41,
92, 80, 2.41,
93, 80, 2.41,
94, 80, -0.586,
95, 80, -1.87,
96, 80, -1.6,
97, 80, -0.52,
98, 80, -0.52,
99, 80, -0.52,
100,    80, -0.288,
101,    80, -0.23,
102,    80, -0.23,
103,    80, 0.292,
104,    80, 2.38,
105,    80, 2.38,
106,    80, 2.38,
107,    80, -0.221,
108,    80, -0.51,
109,    80, -0.51,
110,    80, -0.51,
111,    80, -1.998,
112,    80, -2.37,
113,    80, -2.37,
114,    80, -2.37,
115,    80, -2.37,
116,    80, -2.37,
117,    80, -2.37,
118,    80, -2.37,
119,    80, -1.582,
120,    80, -0.4,
121,    80, -0.4,
122,    80, -0.4,
123,    80, -0.4,
124,    80, -0.4,
125,    80, -0.4,
126,    80, -0.4,
127,    80, 0.52,
128,    80, 0.52,
129,    80, 0.52,
130,    80, 0.52,
131,    80, 0.52,
132,    80, 0.52,
133,    80, -0.472,
134,    80, -1.96,
135,    80, -1.96,
136,    80, -1.006,
137,    80, 1.22,
138,    80, 1.22,
139,    80, 1.131,
140,    80, 0.1,
141,    80, -1.97,
142,    80, -1.97,
143,    80, -1.97,
144,    80, -1.97,
145,    80, -1.97,
146,    80, -1.97,
147,    80, -1.97,
148,    80, -0.745,
149,    80, -0.22,
150,    80, -0.22,
151,    80, -0.036,
152,    80, 0.24,
153,    80, 0.24,
154,    80, 0.24,
155,    80, 0.24,
156,    80, 0.24,
157,    80, 0.09,
158,    80, -0.06,
159,    80, -0.06,
160,    80, -0.06,
161,    80, -0.06,
162,    80, -0.06,
163,    80, 0.868,
164,    80, 1.1,
165,    80, 1.1,
166,    80, 0.456,
167,    80, -0.51,
168,    80, -0.51,
169,    80, -0.51,
170,    80, -0.51,
171,    80, -0.249,
172,    80, 0.36,
173,    80, 0.36,
```

```
174,    80, 0.36,
175,    80, 0.36,
176,    80, 0.36,
177,    80, -0.54,
178,    80, -0.64,
179,    80, -0.122,
180,    80, 1.95,
181,    80, 1.95,
182,    80, 1.68,
183,    80, 1.41,
184,    80, 1.41,
185,    80, 1.41,
186,    80, 1.41,
187,    80, 1.455,
188,    80, 1.86,
189,    80, 1.86,
190,    80, 1.86,
191,    80, 1.86,
192,    80, 1.86,
193,    80, -0.316,
194,    80, -0.86,
195,    80, -0.86,
196,    80, -0.86,
197,    80, -0.86,
198,    80, -1.135,
199,    80, -1.41,
200,    80, -1.41,
201,    80, -1.41,
202,    80, -1.41,
203,    80, -1.239,
204,    80, 0.3,
205,    80, 0.3,
206,    80, 0.126,
207,    80, 0.01,
208,    80, 0.01,
209,    80, 0.01,
210,    80, 0.01,
211,    80, 0.155,
212,    80, 0.3,
213,    80, 0.3,
214,    80, 0.3,
215,    80, 0.76,
216,    80, 2.6,
217,    80, 2.6,
218,    80, 1.3,
219,    80, 1.3,
220,    80, 1.3,
221,    80, 1.3,
222,    80, 1.3,
223,    80, 1.492,
224,    80, 3.22,
225,    80, 3.22,
226,    80, 3.22,
227,    80, 3.22,
228,    80, 3.22,
229,    80, 1.596,
230,    80, 0.9,
231,    80, 0.9,
232,    80, 0.9,
233,    80, 0.9,
234,    80, 0.778,
235,    80, -0.32,
236,    80, -0.32,
237,    80, -2.092,
238,    80, -4.75,
239,    80, -4.75,
240,    80, -4.75,
241,    80, 2.585,
242,    80, 3.4,
243,    80, 2.334,
244,    80, -1.93,
245,    80, -1.93,
246,    80, -1.93,
247,    80, -1.93,
248,    80, -1.93,
249,    80, -1.004,
250,    80, 2.7,
251,    80, 2.7,
252,    80, 2.7,
253,    80, 2.7,
254,    80, 2.97,
255,    80, 3,
256,    80, 3,
257,    80, 3,
258,    80, -4.011,
259,    80, -4.79,
260,    80, -3.546,
261,    80, -1.68,
```

```
262,    80, -1.68,
263,    80, -1.68,
264,    80, -1.68,
265,    80, -1.68,
266,    80, -1.68,
267,    80, -1.68,
268,    80, -1.515,
269,    80, -1.13,
270,    80, -0.757,
271,    80, 2.6,
272,    80, 2.6,
273,    80, -2.144,
274,    80, -2.797,
275,    80, 2.108,
276,    80, 2.54,
277,    80, 3.07,
278,    80, 3.6,
279,    80, 2.652,
280,    80, 0.44,
281,    80, 0.44,
282,    80, -1.926,
283,    80, -2.94,
284,    80, 0.012,
285,    80, 1.53,
286,    80, 1.08,
287,    80, 1.233,
288,    80, 1.25,
289,    80, 1.25,
290,    80, 1.724,
291,    80, 2.04,
292,    80, 2.04,
293,    80, 2.04,
294,    80, 4.74,
295,    80, 5.04,
296,    80, 5.04,
297,    80, 5.04,
298,    80, 5.04,
299,    80, 3.164,
300,    80, 2.36,
301,    80, 2.36,
302,    80, 2.36,
303,    80, 2.36,
304,    80, 1.754,
305,    80, 1.35,
306,    80, 1.35,
307,    80, 1.35,
308,    80, 1.35,
309,    80, 1.35,
310,    80, 1.548,
311,    80, 1.68,
312,    80, 1.68,
313,    80, 1.244,
314,    80, -0.5,
315,    80, -0.5,
316,    80, -0.5,
317,    80, -0.563,
318,    80, -0.71,
319,    80, -0.71,
320,    80, -0.048,
321,    80, 2.6,
322,    80, 2.6,
323,    80, 2.6,
324,    80, 2.6,
325,    80, 2.6,
326,    80, 1.37,
327,    80, 0.55,
328,    80, 0.55,
329,    80, 1.04,
330,    80, 3,
331,    80, 3,
332,    80, 3,
333,    80, 3,
334,    80, 3,
335,    80, 1.12,
336,    80, 1.12,
337,    80, 1.12,
338,    80, 1.12,
339,    80, 1.12,
340,    80, 1.12,
341,    80, 2.88,
342,    80, 2.88,
343,    80, 2.88,
344,    80, 2.88,
345,    80, 2.88,
346,    80, 2.515,
347,    80, -0.77,
348,    80, -0.77,
349,    80, -0.77,
```

```
350,    80, -0.77,
351,    80, -0.77,
352,    80, -0.77,
353,    80, -0.77,
354,    80, -0.77,
355,    80, -0.77,
356,    80, 4,
357,    80, 4,
358,    80, 4,
359,    80, 4,
360,    80, 4,
361,    80, 4,
362,    80, 4,
363,    80, 2.696,
364,    80, 2.37,
365,    80, 2.37,
366,    80, 2.37,
367,    80, -0.126,
368,    80, -0.75,
369,    80, -0.75,
370,    80, -0.75,
371,    80, -0.75,
372,    80, -0.75,
373,    80, -0.51,
374,    80, 1.65,
375,    80, 1.65,
376,    80, 1.347,
377,    80, 0.64,
378,    80, 0.64,
379,    80, 0.64,
380,    80, 0.64,
381,    80, 0.64,
382,    80, -0.18,
383,    80, -1,
384,    80, -1,
385,    80, -1,
386,    80, -0.325,
387,    80, -0.25,
388,    80, -0.25,
389,    80, -0.25,
390,    80, -1.218,
391,    80, -1.46,
392,    80, -1.46,
393,    80, -1.46,
394,    80, -1.46,
395,    80, -1.873,
396,    80, -2.05,
397,    80, -2.05,
398,    80, -2.05,
399,    80, -2.05,
400,    80, -2.05,
401,    80, -2.09,
402,    80, -2.45,
403,    80, -2.45,
404,    80, -2.45,
405,    80, -1.855,
406,    80, -1.26,
407,    80, -1.26,
408,    80, -1.26,
409,    80, -1.26,
410,    80, -1.26,
411,    80, -1.26,
412,    80, -1.26,
413,    80, -1.26,
414,    80, -1.26,
415,    80, -1.26,
416,    80, -1.26,
417,    80, -2.644,
418,    80, -2.99,
419,    80, -2.99,
420,    80, -2.99,
421,    80, -2.99,
422,    80, -2.917,
423,    80, -2.26,
424,    80, -2.26,
425,    80, -2.26,
426,    80, -2.26,
427,    80, -2.26,
428,    80, -2.26,
429,    80, -2.26,
430,    80, -2.26,
431,    80, -2.26,
432,    80, -2.26,
433,    80, -2.26,
434,    80, -2.26,
435,    80, -2.26,
436,    80, -2.26,
437,    80, -2.26,
```

```
438,    80, -2.26,
439,    80, -2.63,
440,    80, -3,
441,    80, -3,
442,    80, -3,
443,    80, -3,
444,    80, -3,
445,    80, -3,
446,    80, 0.77,
447,    80, 0.77,
448,    80, 0.77,
449,    80, 0.77,
450,    80, 0.77,
451,    80, 0.77,
452,    80, 0.77,
453,    80, 0.77,
454,    80, 0.77,
455,    80, 0.77,
456,    80, 0.77,
457,    80, 0.77,
458,    80, 0.149,
459,    80, -1.3,
460,    80, -1.3,
461,    80, -1.3,
462,    80, -1.3,
463,    80, -1.3,
464,    80, -0.18,
465,    80, 0.3,
466,    80, 0.3,
467,    80, 0.3,
468,    80, 0.3,
469,    80, 0.3,
470,    80, 0.3,
471,    80, 0.3,
472,    80, 0.3,
473,    80, 0.3,
474,    80, 0.3,
475,    80, 0.3,
476,    80, 0.3,
477,    80, 0.012,
478,    80, -0.66,
479,    80, -0.66,
480,    80, -0.66,
481,    80, -0.66,
482,    80, -0.866,
483,    80, -2.72,
484,    80, -2.72,
485,    80, -2.72,
486,    80, -2.72,
487,    80, -2.25,
488,    80, -0.37,
489,    80, -0.37,
490,    80, -0.37,
491,    80, -0.37,
492,    80, -0.37,
493,    80, -0.37,
494,    80, -0.37,
495,    80, -0.37,
496,    80, -0.632,
497,    80, -1.68,
498,    80, -1.68,
499,    80, -1.68,
500,    80, -1.68,
501,    80, -1.68,
502,    80, -1.68,
503,    80, -1.68,
504,    80, -1.68,
505,    80, -0.154,
506,    80, 0.5,
507,    80, 0.5,
508,    80, 0.5,
509,    80, 0.5,
510,    80, 0.5,
511,    80, 0.5,
512,    80, -0.217,
513,    80, -1.89,
514,    80, -1.89,
515,    80, -1.89,
516,    80, -1.89,
517,    80, 1.251,
518,    80, 1.6,
519,    80, 1.6,
520,    80, 1.6,
521,    80, 1.6,
522,    80, 1.6,
523,    80, 1.222,
524,    80, -0.29,
525,    80, -0.29,
```

```
526,    80, -0.29,
527,    80, -0.29,
528,    80, -0.29,
529,    80, 1.02,
530,    80, 2.33,
531,    80, 2.33,
532,    80, 2.33,
533,    80, 2.33,
534,    80, -0.03,
535,    80, -0.62,
536,    80, -0.62,
537,    80, -0.62,
538,    80, 0.148,
539,    80, 1.3,
540,    80, 1.3,
541,    80, 1.3,
542,    80, 0.6,
543,    80, -0.45,
544,    80, -0.45,
545,    80, -0.45,
546,    80, -0.208,
547,    80, 0.76,
548,    80, 0.76,
549,    80, 0.76,
550,    80, 0.76,
551,    80, 0.76,
552,    80, 0.76,
553,    80, 0.76,
554,    80, 0.76,
555,    80, 0.76,
556,    80, -2.76,
557,    80, -2.76,
558,    80, -2.76,
559,    80, -2.76,
560,    80, -2.76,
561,    80, -2.76,
562,    80, -2.76,
563,    80, -2.76,
564,    80, -2.76,
565,    80, -2.76,
566,    80, -2.76,
567,    80, -2.76,
568,    80, -2.76,
569,    80, -0.192,
570,    80, 0.45,
571,    80, 0.45,
572,    80, 0.45,
573,    80, 0.88,
574,    80, 2.6,
575,    80, 2.6,
576,    80, 2.6,
577,    80, 2.6,
578,    80, 2.6,
579,    80, 2.6,
580,    80, 2.6,
581,    80, 2.6,
582,    80, 2.6,
583,    80, 0.5,
584,    80, -2.65,
585,    80, -2.65,
586,    80, -2.65,
587,    80, -2.65,
588,    80, -2.65,
589,    80, -2.65,
590,    80, -2.65,
591,    80, -2.65,
592,    80, -2.65,
593,    80, -2.335,
594,    80, 0.5,
595,    80, 0.5,
596,    80, 0.5,
597,    80, 0.5,
598,    80, 0.5,
599,    80, 0.5,
600,    80, 0.5,
601,    80, 0.05,
602,    80, -0.4,
603,    80, -0.4,
604,    80, -0.4,
605,    80, -0.4,
606,    80, -0.4,
607,    80, -0.4,
608,    80, -0.4,
609,    80, -1.04,
610,    80, -2,
611,    80, -2,
612,    80, -1.8,
613,    80, 0,
```

```
614,    80, 0,
615,    80, 0,
616,    80, 0.15,
617,    80, 0.5,
618,    80, 0.5,
619,    80, 0.5,
620,    80, 0.248,
621,    80, -0.76,
622,    80, -0.76,
623,    80, -0.76,
624,    80, -0.76,
625,    80, -0.76,
626,    80, -0.76,
627,    80, -0.76,
628,    80, -0.76,
629,    80, -0.228,
630,    80, 0,
631,    80, 0,
632,    80, 0,
633,    80, 0,
634,    80, 2.232,
635,    80, 2.79,
636,    80, 2.79,
637,    80, 2.79,
638,    80, 2.79,
639,    80, 1.772,
640,    80, -2.3,
641,    80, -2.3,
642,    80, -2.3,
643,    80, -2.3,
644,    80, -2.3,
645,    80, -2.3,
646,    80, -2.3,
647,    80, -2.3,
648,    80, 0.962,
649,    80, 2.36,
650,    80, 2.36,
651,    80, 2.36,
652,    80, 2.36,
653,    80, 2.36,
654,    80, 2.36,
655,    80, 2.36,
656,    80, 2.36,
657,    80, -0.692,
658,    80, -2,
659,    80, -2,
660,    80, -2,
661,    80, 0.403,
662,    80, 0.67,
663,    80, 0.67,
664,    80, 0.67,
665,    80, -0.257,
666,    80, -0.36,
667,    80, -0.36,
668,    80, -0.36,
669,    80, -0.36,
670,    80, -0.297,
671,    80, -0.29,
672,    80, -0.29,
673,    80, -0.29,
674,    80, -0.29,
675,    80, -0.346,
676,    80, -0.37,
677,    80, -0.37,
678,    80, -0.37,
679,    80, -0.37,
680,    80, -0.37,
681,    80, -0.37,
682,    80, -0.37,
683,    80, -0.37,
684,    80, -0.37,
685,    80, -0.37,
686,    80, -0.37,
687,    80, -0.37,
688,    80, -0.37,
689,    80, -0.37,
690,    80, 0.08,
691,    80, 0.13,
692,    80, 0.13,
693,    80, 0.13,
694,    80, 0.13,
695,    80, 0.13,
696,    80, 0.13,
697,    80, 1.23,
698,    80, 2.33,
699,    80, 2.33,
700,    80, 2.33,
701,    80, 2.33,
```

```
702,    80, 2.33,
703,    80, 2.33,
704,    80, 2.33,
705,    80, 2.33,
706,    80, 2.33,
707,    80, 2.33,
708,    80, -0.541,
709,    80, -0.86,
710,    80, -0.86,
711,    80, -0.86,
712,    80, -0.86,
713,    80, -1.105,
714,    80, -1.35,
715,    80, -1.35,
716,    80, -1.35,
717,    80, -1.35,
718,    80, -1.35,
719,    80, -1.35,
720,    80, -1.35,
721,    80, -1.35,
722,    80, -1.35,
723,    80, -1.35,
724,    80, -1.35,
725,    80, -0.618,
726,    80, -0.13,
727,    80, -0.13,
728,    80, 0.293,
729,    80, 0.34,
730,    80, 0.34,
731,    80, 0.34,
732,    80, 0.34,
733,    80, 0.34,
734,    80, 0.34,
735,    80, 0.34,
736,    80, 0.34,
737,    80, -0.045,
738,    80, -0.43,
739,    80, -0.43,
740,    80, -0.43,
741,    80, -0.43,
742,    80, -0.43,
743,    80, 0.102,
744,    80, 0.9,
745,    80, 0.9,
746,    80, 0.9,
747,    80, 0.366,
748,    80, -0.88,
749,    80, -0.88,
750,    80, -0.88,
751,    80, -0.88,
752,    80, -0.88,
753,    80, -0.88,
754,    80, -0.88,
755,    80, -0.88,
756,    80, 0.208,
757,    80, 1.84,
758,    80, 1.84,
759,    80, 1.84,
760,    80, 1.84,
761,    80, 1.84,
762,    80, 1.84,
763,    80, 1.84,
764,    80, 1.84,
765,    80, 1.406,
766,    80, -2.5,
767,    80, -2.5,
768,    80, -2.5,
769,    80, -2.5,
770,    80, -2.5,
771,    80, -0.1,
772,    80, -0.1,
773,    80, -0.1,
774,    80, -0.1,
775,    80, -0.1,
776,    80, -0.1,
777,    80, -0.1,
778,    80, -0.1,
779,    80, -0.1,
780,    80, -0.1,
781,    80, 0.14,
782,    80, 0.2,
783,    80, 0.2,
784,    80, 0.2,
785,    80, 0.2,
786,    80, 0.22,
787,    80, 0.3,
788,    80, 0.3,
789,    80, 0.3,
```

```
790,    80, 0.25,
791,    80, -0.2,
792,    80, -0.2,
793,    80, -0.2,
794,    80, -0.2,
795,    80, -0.2,
796,    80, 1.24,
797,    80, 1.6,
798,    80, 1.6,
799,    80, 1.24,
800,    80, -2,
801,    80, -2,
802,    80, -2,
803,    80, -2,
804,    80, -2,
805,    80, -2,
806,    80, -2,
807,    80, -2,
808,    80, -2,
809,    80, -0.02,
810,    80, 0.2,
811,    80, 0.2,
812,    80, 0.2,
813,    80, 0.2,
814,    80, 0.2,
815,    80, 0.2,
816,    80, 0.083,
817,    80, 0.07,
818,    80, 0.07,
819,    80, 0.07,
820,    80, 0.07,
821,    80, 0.07,
822,    80, 0.07,
823,    80, 0.07,
824,    80, 0.222,
825,    80, 0.45,
826,    80, 0.45,
827,    80, 0.45,
828,    80, 0.45,
829,    80, 0.45,
830,    80, 0.45,
831,    80, 0.45,
832,    80, 0.45,
833,    80, 0.45,
834,    80, 0.385,
835,    80, -0.2,
836,    80, -0.2,
837,    80, -0.2,
838,    80, -0.2,
839,    80, -0.2,
840,    80, -0.2,
841,    80, -0.2,
842,    80, -0.2,
843,    80, -0.2,
844,    80, -0.398,
845,    80, -2.18,
846,    80, -2.18,
847,    80, -2.18,
848,    80, -2.18,
849,    80, 1.048,
850,    80, 3.2,
851,    80, 3.2,
852,    80, 3.2,
853,    80, 3.2,
854,    80, 1.555,
855,    80, 0.85,
856,    80, 0.85,
857,    80, 0.85,
858,    80, -1.09,
859,    80, -4,
860,    80, -4,
861,    80, -4,
862,    80, -4,
863,    80, -4,
864,    80, -4,
865,    80, -4,
866,    80, -4,
867,    80, -4,
868,    80, -4,
869,    80, 2.8,
870,    80, 4.5,
871,    80, 4.5,
872,    80, 4.5,
873,    80, 4.5,
874,    80, 4.5,
875,    80, 4.5,
876,    80, 2.5,
877,    80, 2,
```

```
878,    80, 2,
879,    80, 2,
880,    80, 2,
881,    80, 2,
882,    80, 2,
883,    80, 2,
884,    80, 2,
885,    80, 1.686,
886,    80, 0.43,
887,    80, 0.43,
888,    80, 0.43,
889,    80, 0.43,
890,    80, 0.43,
891,    80, 0.43,
892,    80, 0.43,
893,    80, -1.714,
894,    80, -4.93,
895,    80, -4.93,
896,    80, -4.93,
897,    80, -4.93,
898,    80, -4.93,
899,    80, -2.059,
900,    80, -1.74,
901,    80, -1.74,
902,    80, -1.74,
903,    80, -1.74,
904,    80, -2.37,
905,    80, -3,
906,    80, -3,
907,    80, -2.719,
908,    80, -0.19,
909,    80, -0.19,
910,    80, -0.19,
911,    80, -0.19,
912,    80, -0.19,
913,    80, -0.19,
914,    80, -0.19,
915,    80, -0.19,
916,    80, 1.84,
917,    80, 1.84,
918,    80, 1.84,
919,    80, 1.84,
920,    80, 1.732,
921,    80, 0.76,
922,    80, 0.76,
923,    80, 0.76,
924,    80, -0.423,
925,    80, -0.93,
926,    80, -0.93,
927,    80, -0.604,
928,    80, 0.7,
929,    80, 0.7,
930,    80, 0.7,
931,    80, 0.7,
932,    80, -0.92,
933,    80, -1.1,
934,    80, -1.1,
935,    80, -1.1,
936,    80, -1.1,
937,    80, -0.828,
938,    80, -0.42,
939,    80, -0.42,
940,    80, -0.42,
941,    80, -0.42,
942,    80, -0.42,
943,    80, -0.42,
944,    80, -0.42,
945,    80, -0.078,
946,    80, 0.72,
947,    80, 0.72,
948,    80, 0.72,
949,    80, -0.37,
950,    80, -0.37,
951,    80, -0.37,
952,    80, -0.37,
953,    80, -0.812,
954,    80, -2.58,
955,    80, -2.58,
956,    80, -2.58,
957,    80, -2.58,
958,    80, -2.58,
959,    80, -2.58,
960,    80, -2.58,
961,    80, -2.58,
962,    80, -2.58,
963,    80, -2.58,
964,    80, -2.58,
965,    80, -2.58,
```

```
966,    80, -0.683,
967,    80, 0.13,
968,    80, 0.13,
969,    80, 0.13,
970,    80, 0.13,
971,    80, 0.13,
972,    80, 0.13,
973,    80, 0.13,
974,    80, -0.493,
975,    80, -0.76,
976,    80, -0.76,
977,    80, -0.76,
978,    80, -0.76,
979,    80, -0.76,
980,    80, -0.76,
981,    80, -0.76,
982,    80, -0.394,
983,    80, 0.46,
984,    80, 0.46,
985,    80, 0.46,
986,    80, 0.46,
987,    80, -0.317,
988,    80, -0.65,
989,    80, -0.65,
990,    80, -0.65,
991,    80, -0.65,
992,    80, -0.65,
993,    80, -0.65,
994,    80, -0.65,
995,    80, 0.136,
996,    80, 0.66,
997,    80, 0.66,
998,    80, 0.66,
999,    80, 0.66,
1000,   80, 0.66,
1001,   80, 3,
1002,   80, 3,
1003,   80, 3,
1004,   80, 3,
1005,   80, 3,
1006,   80, 1.068,
1007,   80, -1.83,
1008,   80, -1.83,
1009,   80, -1.83,
1010,   80, -1.83,
1011,   80, -1.83,
1012,   80, 0.126,
1013,   80, 1.43,
1014,   80, 1.43,
1015,   80, 1.43,
1016,   80, 1.43,
1017,   80, 0.576,
1018,   80, -2.84,
1019,   80, -2.84,
1020,   80, -2.84,
1021,   80, -2.84,
1022,   80, -2.84,
1023,   80, -2.84,
1024,   80, -1.27,
1025,   80, 0.3,
1026,   80, 0.3,
1027,   80, 0.3,
1028,   80, 0.3,
1029,   80, 0.3,
1030,   80, 0.3,
1031,   80, -0.51,
1032,   80, -0.51,
1033,   80, -0.51,
1034,   80, -0.51,
1035,   80, 0.2,
1036,   80, 0.2,
1037,   80, 0.2,
1038,   80, 0.2,
1039,   80, 0.2,
1040,   80, 0.2,
1041,   80, -0.087,
1042,   80, -0.21,
1043,   80, -0.21,
1044,   80, -0.21,
1045,   80, 0.036,
1046,   80, 0.2,
1047,   80, 0.2,
1048,   80, 0.2,
1049,   80, 0.06,
1050,   80, 0,
1051,   80, 0,
1052,   80, 0,
1053,   80, -0.12,
```

```
1054,   80, -0.4,
1055,   80, -0.4,
1056,   80, -0.4,
1057,   80, -0.4,
1058,   80, 0.8,
1059,   80, 2.6,
1060,   80, 2.6,
1061,   80, 2.6,
1062,   80, 2.6,
1063,   80, 2.6,
1064,   80, 2.6,
1065,   80, 2.6,
1066,   80, 1.42,
1067,   80, 0.24,
1068,   80, 0.24,
1069,   80, 1.235,
1070,   80, 2.23,
1071,   80, 2.23,
1072,   80, 2.23,
1073,   80, 2.23,
1074,   80, 2.23,
1075,   80, 2.23,
1076,   80, 2.23,
1077,   80, 2.23,
1078,   80, 2.23,
1079,   80, 2.23,
1080,   80, 2.23,
1081,   80, 2.23,
1082,   80, 0.529,
1083,   80, -0.2,
1084,   80, -0.2,
1085,   80, -0.2,
1086,   80, -0.2,
1087,   80, -0.2,
1088,   80, -0.76,
1089,   80, -3,
1090,   80, -3,
1091,   80, -3,
1092,   80, 0.21,
1093,   80, 2.35,
1094,   80, 2.35,
1095,   80, 2.35,
1096,   80, 2.35,
1097,   80, 0.541,
1098,   80, 0.34,
1099,   80, 0.34,
1100,   80, 0.34,
1101,   80, -0.17,
1102,   80, -0.51,
1103,   80, -0.51,
1104,   80, -0.51,
1105,   80, -1.302,
1106,   80, -1.83,
1107,   80, -1.83,
1108,   80, -1.83,
1109,   80, -1.83,
1110,   80, -1.83,
1111,   80, -1.83,
1112,   80, -1.83,
1113,   80, -1.83,
1114,   80, -0.006,
1115,   80, -0.302,
1116,   80, -1.43,
1117,   80, -1.43,
1118,   80, -1.43,
1119,   80, -1.1,
1120,   80, -1.1,
1121,   80, -1.1,
1122,   80, -1.1,
1123,   80, 0.3,
1124,   80, 1.7,
1125,   80, 1.7,
1126,   80, 1.7,
1127,   80, 1.7,
1128,   80, 1.7,
1129,   80, 1.228,
1130,   80, -0.66,
1131,   80, -0.66,
1132,   80, -0.66,
1133,   80, -0.66,
1134,   80, 0.339,
1135,   80, 2.67,
1136,   80, 2.67,
1137,   80, 0.674,
1138,   80, -2.32,
1139,   80, -2.32,
1140,   80, -1.356,
1141,   80, 2.5,
```

```
1142,   80, 2.5,
1143,   80, 2.5,
1144,   80, 2.5,
1145,   80, -1,
1146,   80, -2.5,
1147,   80, -2.5,
1148,   80, -2.5,
1149,   80, -2.5,
1150,   80, -2.5,
1151,   80, -2.5,
1152,   80, -2.5,
1153,   80, -2.5,
1154,   80, -2.5,
1155,   80, -2.5,
1156,   80, -2.5,
1157,   80, -2.09,
1158,   80, 1.6,
1159,   80, 1.6,
1160,   80, 1.6,
1161,   80, 1.6,
1162,   80, 1.6,
1163,   80, 2.23,
1164,   80, 2.5,
1165,   80, 2.5,
1166,   80, 1.573,
1167,   80, -0.59,
1168,   80, -0.533,
1169,   80, -0.4,
1170,   80, -0.4,
1171,   80, -0.4,
1172,   80, 0.24,
1173,   80, 0.4,
1174,   80, 0.4,
1175,   80, 0.22,
1176,   80, -0.5,
1177,   80, -0.5,
1178,   80, -0.968,
1179,   80, -1.67,
1180,   80, -1.67,
1181,   80, -1.67,
1182,   80, -1.67,
1183,   80, -0.985,
1184,   80, -0.3,
1185,   80, -0.96,
1186,   80, -2.5,
1187,   80, -2.22,
1188,   80, -1.8,
1189,   80, -1.76,
1190,   80, -1.6,
1191,   80, -1.6,
1192,   80, -1.6,
1193,   80, -1.6,
1194,   80, -1.6,
1195,   80, -1.6,
1196,   80, -1.6,
1197,   80, -1.6,
1198,   80, -0.82,
1199,   80, -0.3,
1200,   80, -1.92,
1201,   80, -3,
1202,   80, -3,
1203,   80, -3,
1204,   80, -3,
1205,   80, -3,
1206,   80, -3,
1207,   80, -0.282,
1208,   80, 0.02,
1209,   80, 0.02,
1210,   80, 1.155,
1211,   80, 2.11,
1212,   80, 2.11,
1213,   80, 1.805,
1214,   80, -0.94,
1215,   80, -0.772,
1216,   80, 0.18,
1217,   80, 0.25,
1218,   80, 0.25,
1219,   80, 0.25,
1220,   80, 0.4,
1221,   80, 0.58,
1222,   80, 1.3,
1223,   80, 0.8,
1224,   80, 0.3,
1225,   80, 1.2,
1226,   80, 1.8,
1227,   80, 1.64,
1228,   80, 1,
1229,   80, 1,
```

```
1230,   80, -0.4,
1231,   80, -1,
1232,   80, -0.28,
1233,   80, -0.2,
1234,   80, -0.2,
1235,   80, -0.2,
1236,   80, -0.08,
1237,   80, 1,
1238,   80, 1,
1239,   80, 1,
1240,   80, 1,
1241,   80, -1.24,
1242,   80, -2.2,
1243,   80, -2.2,
1244,   80, -2.2,
1245,   80, 1.8,
1246,   80, 1.8,
1247,   80, 1.8,
1248,   80, 1.8,
1249,   80, 1.8,
1250,   80, 2.79,
1251,   80, 2.9,
1252,   80, 2.9,
1253,   80, 2.9,
1254,   80, 1.72,
1255,   80, -3,
1256,   80, -3,
1257,   80, -3,
1258,   80, 1,
1259,   80, 1,
1260,   80, 1.8,
1261,   80, 3,
1262,   80, 3,
1263,   80, 1.5,
1264,   80, 0.5,
1265,   80, 0.92,
1266,   80, 1.2,
1267,   80, 1.28,
1268,   80, 1.6,
1269,   80, 1.6,
1270,   80, 2.72,
1271,   80, 3,
1272,   80, 1.945,
1273,   80, 0.89,
1274,   80, 0.861,
1275,   80, 0.6,
1276,   80, 0.6,
1277,   80, 0.6,
1278,   80, 1.288,
1279,   80, 1.46,
1280,   80, -0.67,
1281,   80, -2.8,
1282,   80, -2.8,
1283,   80, -2.8,
1284,   80, -2.8,
1285,   80, -2.06,
1286,   80, -1.32,
1287,   80, -1.101,
1288,   80, 0.87,
1289,   80, 0.87,
1290,   80, -0.544,
1291,   80, -1.15,
1292,   80, -1.15,
1293,   80, -1.42,
1294,   80, -1.45,
1295,   80, -1.765,
1296,   80, -1.9,
1297,   80, -1.9,
1298,   80, -1.9,
1299,   80, -1.9,
1300,   80, -0.6,
1301,   80, -0.6,
1302,   80, -0.744,
1303,   80, -0.76,
1304,   80, -0.76,
1305,   80, -1.23,
1306,   80, -1.23,
1307,   80, -1.23,
1308,   80, -1.23,
1309,   80, -1.23,
1310,   80, -1.23,
1311,   80, -1.23,
1312,   80, -1.23,
1313,   80, 0.246,
1314,   80, 0.41,
1315,   80, 0.978,
1316,   80, 1.12,
1317,   80, 1.12,
```

```
1318,   80, 1.12,
1319,   80, 1.12,
1320,   80, -0.608,
1321,   80, -0.8,
1322,   80, -0.02,
1323,   80, 0.6,
1324,   80, 1,
1325,   80, 1,
1326,   80, 1.9,
1327,   80, 2,
1328,   80, 2,
1329,   80, 2,
1330,   80, 0.95,
1331,   80, -1.5,
1332,   80, -1.35,
1333,   80, -0.75,
1334,   80, -0.75,
1335,   80, -0.75,
1336,   80, -0.75,
1337,   80, -0.75,
1338,   80, -0.675,
1339,   80, -0.5,
1340,   80, -0.5,
1341,   80, -0.3,
1342,   80, 1.5,
1343,   80, 1.5,
1344,   80, 1.5,
1345,   80, 1.5,
1346,   80, 1.5,
1347,   80, 1.95,
1348,   80, 2,
1349,   80, 0.6,
1350,   80, -1.5,
1351,   80, -1.5,
1352,   80, 1.2,
1353,   80, 3,
1354,   80, 1.1,
1355,   80, -0.8,
1356,   80, -0.98,
1357,   80, -1.1,
1358,   80, -2,
1359,   80, -2,
1360,   80, -2,
1361,   80, -2,
1362,   80, -2,
1363,   80, -0.926,
1364,   80, -0.21,
1365,   80, -0.21,
1366,   80, -0.21,
1367,   80, -0.912,
1368,   80, -0.99,
1369,   80, -0.621,
1370,   80, 0.24,
1371,   80, 0.24,
1372,   80, 0.966,
1373,   80, 1.45,
1374,   80, 1.45,
1375,   80, 0.89,
1376,   80, 0.05,
1377,   80, 0.05,
1378,   80, 0.05,
1379,   80, 0.05,
1380,   80, 0.05,
1381,   80, 0.995,
1382,   80, 1.1,
1383,   80, 1.1,
1384,   80, -1.412,
1385,   80, -2.04,
1386,   80, -2.04,
1387,   80, -2.04,
1388,   80, -2.04,
1389,   80, -0.87,
1390,   80, 0.3,
1391,   80, 0.3,
1392,   80, 0.3,
1393,   80, 0.3,
1394,   80, 0.3,
1395,   80, 0.3,
1396,   80, -0.18,
1397,   80, -0.3,
1398,   80, -0.295,
1399,   80, -0.29,
1400,   80, -0.29,
1401,   80, -0.29,
1402,   80, -0.29,
1403,   80, -0.29,
1404,   80, -0.29,
1405,   80, -0.29,
```

```
1406,   80, -0.29,
1407,   80, -0.29,
1408,   80, 2.671,
1409,   80, 3,
1410,   80, -0.97,
1411,   80, -0.97,
1412,   80, -0.97,
1413,   80, -0.97,
1414,   80, -0.97,
1415,   80, -0.97,
1416,   80, -0.97,
1417,   80, -0.97,
1418,   80, -0.97,
1419,   80, -0.97,
1420,   80, -0.97,
1421,   80, -0.97,
1422,   80, -0.182,
1423,   80, 1,
1424,   80, 1,
1425,   80, 1,
1426,   80, 1.6,
1427,   80, 2,
1428,   80, 2,
1429,   80, 1.04,
1430,   80, 0.4,
1431,   80, 0.128,
1432,   80, -0.28,
1433,   80, -0.28,
1434,   80, -0.28,
1435,   80, -0.772,
1436,   80, -1.1,
1437,   80, -1.1,
1438,   80, -1.1,
1439,   80, -1.1,
1440,   80, 0.35,
1441,   80, 0.35,
1442,   80, 0.35,
1443,   80, 0.35,
1444,   80, -1.63,
1445,   80, -1.63,
1446,   80, -1.63,
1447,   80, -1.365,
1448,   80, -1.1,
1449,   80, -1.1,
1450,   80, -1.1,
1451,   80, 0.85,
1452,   80, 2.8,
1453,   80, 2.8,
1454,   80, 2.745,
1455,   80, 2.25,
1456,   80, 2.25,
1457,   80, 2.25,
1458,   80, 1.34,
1459,   80, -2.3,
1460,   80, -2.3,
1461,   80, -2.18,
1462,   80, -2,
1463,   80, -2,
1464,   80, -0.536,
1465,   80, 0.6,
1466,   80, 0.6,
1467,   80, 0.6,
1468,   80, 0.6,
1469,   80, 0.6,
1470,   80, 0.6,
1471,   80, 0.6,
1472,   80, 0.6,
1473,   80, 0.6,
1474,   80, 0.6,
1475,   80, -0.84,
1476,   80, -0.84,
1477,   80, -0.144,
1478,   80, 0.9,
1479,   80, 0.9,
1480,   80, 0.9,
1481,   80, 0.9,
1482,   80, 0.9,
1483,   80, 0.9,
1484,   80, -0.189,
1485,   80, -0.31,
1486,   80, -0.31,
1487,   80, -0.31,
1488,   80, -0.31,
1489,   80, -0.31,
1490,   80, 0.683,
1491,   80, 3,
1492,   80, 3,
1493,   80, 3,
```

```
1494,   80, 3,
1495,   80, 3,
1496,   80, 3,
1497,   80, 3,
1498,   80, 3,
1499,   80, 3,
1500,   80, 3,
1501,   80, 3,
1502,   80, 3,
1503,   80, 2,
1504,   80, -2,
1505,   80, -2,
1506,   80, -2,
1507,   80, -2,
1508,   80, -2,
1509,   80, 0.8,
1510,   80, 2,
1511,   80, 2,
1512,   80, 2,
1513,   80, 0.144,
1514,   80, -0.32,
1515,   80, -0.32,
1516,   80, -0.32,
1517,   80, 0.608,
1518,   80, 2,
1519,   80, 0.512,
1520,   80, 0.14,
1521,   80, 0.14,
1522,   80, 0.14,
1523,   80, 0.14,
1524,   80, 0.14,
1525,   80, 0.14,
1526,   80, 0.14,
1527,   80, 0.14,
1528,   80, 0.14,
1529,   80, 1.814,
1530,   80, 2,
1531,   80, 2,
1532,   80, 2,
1533,   80, 2,
1534,   80, 2,
1535,   80, 2,
1536,   80, 2,
1537,   80, 2,
1538,   80, 2,
1539,   80, 2,
1540,   80, 0.215,
1541,   80, -1.57,
1542,   80, -1.57,
1543,   80, -1.57,
1544,   80, -1.57,
1545,   80, -1.57,
1546,   80, -1.57,
1547,   80, -1.57,
1548,   80, -0.331,
1549,   80, 0.2,
1550,   80, 0.2,
1551,   80, -0.25,
1552,   80, -0.7,
1553,   80, -0.63,
1554,   80, 0,
1555,   80, 0,
1556,   80, 0,
1557,   80, 0,
1558,   80, 0,
1559,   80, -0.414,
1560,   80, 0.414,
1561,   80, 0,
1562,   80, 0,
1563,   80, 0,
1564,   80, 0,
1565,   80, 0,
1566,   80, 0.63,
1567,   80, 0.7,
1568,   80, 0.25,
1569,   80, -0.2,
1570,   80, -0.2,
1571,   80, 0.331,
1572,   80, 1.57,
1573,   80, 1.57,
1574,   80, 1.57,
1575,   80, 1.57,
1576,   80, 1.57,
1577,   80, 1.57,
1578,   80, 1.57,
1579,   80, -0.215,
1580,   80, -2,
1581,   80, -2,
```

```
1582,   80, -2,
1583,   80, -2,
1584,   80, -2,
1585,   80, -2,
1586,   80, -2,
1587,   80, -2,
1588,   80, -2,
1589,   80, -2,
1590,   80, -1.814,
1591,   80, -0.14,
1592,   80, -0.14,
1593,   80, -0.14,
1594,   80, -0.14,
1595,   80, -0.14,
1596,   80, -0.14,
1597,   80, -0.14,
1598,   80, -0.14,
1599,   80, -0.14,
1600,   80, -0.512,
1601,   80, -2,
1602,   80, -0.608,
1603,   80, 0.32,
1604,   80, 0.32,
1605,   80, 0.32,
1606,   80, -0.144,
1607,   80, -2,
1608,   80, -2,
1609,   80, -2,
1610,   80, -0.8,
1611,   80, 2,
1612,   80, 2,
1613,   80, 2,
1614,   80, 2,
1615,   80, 2,
1616,   80, -2,
1617,   80, -3,
1618,   80, -3,
1619,   80, -3,
1620,   80, -3,
1621,   80, -3,
1622,   80, -3,
1623,   80, -3,
1624,   80, -3,
1625,   80, -3,
1626,   80, -3,
1627,   80, -3,
1628,   80, -3,
1629,   80, -0.683,
1630,   80, 0.31,
1631,   80, 0.31,
1632,   80, 0.31,
1633,   80, 0.31,
1634,   80, 0.31,
1635,   80, 0.189,
1636,   80, -0.9,
1637,   80, -0.9,
1638,   80, -0.9,
1639,   80, -0.9,
1640,   80, -0.9,
1641,   80, -0.9,
1642,   80, 0.144,
1643,   80, 0.84,
1644,   80, 0.84,
1645,   80, -0.6,
1646,   80, -0.6,
1647,   80, -0.6,
1648,   80, -0.6,
1649,   80, -0.6,
1650,   80, -0.6,
1651,   80, -0.6,
1652,   80, -0.6,
1653,   80, -0.6,
1654,   80, -0.6,
1655,   80, 0.536,
1656,   80, 2,
1657,   80, 2,
1658,   80, 2.18,
1659,   80, 2.3,
1660,   80, 2.3,
1661,   80, -1.34,
1662,   80, -2.25,
1663,   80, -2.25,
1664,   80, -2.25,
1665,   80, -2.745,
1666,   80, -2.8,
1667,   80, -2.8,
1668,   80, -0.85,
1669,   80, 1.1,
```

```
1670,   80, 1.1,
1671,   80, 1.1,
1672,   80, 1.365,
1673,   80, 1.63,
1674,   80, 1.63,
1675,   80, 1.63,
1676,   80, -0.35,
1677,   80, -0.35,
1678,   80, -0.35,
1679,   80, -0.35,
1680,   80, 1.1,
1681,   80, 1.1,
1682,   80, 1.1,
1683,   80, 1.1,
1684,   80, 0.772,
1685,   80, 0.28,
1686,   80, 0.28,
1687,   80, 0.28,
1688,   80, -0.128,
1689,   80, -0.4,
1690,   80, -1.04,
1691,   80, -2,
1692,   80, -2,
1693,   80, -1.6,
1694,   80, -1,
1695,   80, -1,
1696,   80, -1,
1697,   80, 0.182,
1698,   80, 0.97,
1699,   80, 0.97,
1700,   80, 0.97,
1701,   80, 0.97,
1702,   80, 0.97,
1703,   80, 0.97,
1704,   80, 0.97,
1705,   80, 0.97,
1706,   80, 0.97,
1707,   80, 0.97,
1708,   80, 0.97,
1709,   80, 0.97,
1710,   80, -3,
1711,   80, -2.671,
1712,   80, 0.29,
1713,   80, 0.29,
1714,   80, 0.29,
1715,   80, 0.29,
1716,   80, 0.29,
1717,   80, 0.29,
1718,   80, 0.29,
1719,   80, 0.29,
1720,   80, 0.29,
1721,   80, 0.295,
1722,   80, 0.3,
1723,   80, 0.18,
1724,   80, -0.3,
1725,   80, -0.3,
1726,   80, -0.3,
1727,   80, -0.3,
1728,   80, -0.3,
1729,   80, -0.3,
1730,   80, 0.87,
1731,   80, 2.04,
1732,   80, 2.04,
1733,   80, 2.04,
1734,   80, 2.04,
1735,   80, 1.412,
1736,   80, -1.1,
1737,   80, -1.1,
1738,   80, -0.995,
1739,   80, -0.05,
1740,   80, -0.05,
1741,   80, -0.05,
1742,   80, -0.05,
1743,   80, -0.05,
1744,   80, -0.89,
1745,   80, -1.45,
1746,   80, -1.45,
1747,   80, -0.966,
1748,   80, -0.24,
1749,   80, -0.24,
1750,   80, 0.621,
1751,   80, 0.99,
1752,   80, 0.912,
1753,   80, 0.21,
1754,   80, 0.21,
1755,   80, 0.21,
1756,   80, 0.926,
1757,   80, 2,
```

```
1758,   80, 2,
1759,   80, 2,
1760,   80, 2,
1761,   80, 2,
1762,   80, 1.1,
1763,   80, 0.98,
1764,   80, 0.8,
1765,   80, -1.1,
1766,   80, -3,
1767,   80, -1.2,
1768,   80, 1.5,
1769,   80, 1.5,
1770,   80, -0.6,
1771,   80, -2,
1772,   80, -1.95,
1773,   80, -1.5,
1774,   80, -1.5,
1775,   80, -1.5,
1776,   80, -1.5,
1777,   80, -1.5,
1778,   80, 0.3,
1779,   80, 0.5,
1780,   80, 0.5,
1781,   80, 0.675,
1782,   80, 0.75,
1783,   80, 0.75,
1784,   80, 0.75,
1785,   80, 0.75,
1786,   80, 0.75,
1787,   80, 1.35,
1788,   80, 1.5,
1789,   80, -0.95,
1790,   80, -2,
1791,   80, -2,
1792,   80, -2,
1793,   80, -1.9,
1794,   80, -1,
1795,   80, -1,
1796,   80, -0.6,
1797,   80, 0.02,
1798,   80, 0.8,
1799,   80, 0.608,
1800,   80, -1.12,
1801,   80, -1.12,
1802,   80, -1.12,
1803,   80, -1.12,
1804,   80, -0.978,
1805,   80, -0.41,
1806,   80, -0.246,
1807,   80, 1.23,
1808,   80, 1.23,
1809,   80, 1.23,
1810,   80, 1.23,
1811,   80, 1.23,
1812,   80, 1.23,
1813,   80, 1.23,
1814,   80, 1.23,
1815,   80, 0.76,
1816,   80, 0.76,
1817,   80, 0.744,
1818,   80, 0.6,
1819,   80, 0.6,
1820,   80, 1.9,
1821,   80, 1.9,
1822,   80, 1.9,
1823,   80, 1.9,
1824,   80, 1.765,
1825,   80, 1.45,
1826,   80, 1.42,
1827,   80, 1.15,
1828,   80, 1.15,
1829,   80, 0.544,
1830,   80, -0.87,
1831,   80, -0.87,
1832,   80, 1.101,
1833,   80, 1.32,
1834,   80, 2.06,
1835,   80, 2.8,
1836,   80, 2.8,
1837,   80, 2.8,
1838,   80, 2.8,
1839,   80, 0.67,
1840,   80, -1.46,
1841,   80, -1.288,
1842,   80, -0.6,
1843,   80, -0.6,
1844,   80, -0.6,
1845,   80, -0.861,
```

```
1846,   80, -0.89,
1847,   80, -1.945,
1848,   80, -3,
1849,   80, -2.72,
1850,   80, -1.6,
1851,   80, -1.6,
1852,   80, -1.28,
1853,   80, -1.2,
1854,   80, -0.92,
1855,   80, -0.5,
1856,   80, -1.5,
1857,   80, -3,
1858,   80, -3,
1859,   80, -1.8,
1860,   80, -1,
1861,   80, -1,
1862,   80, 3,
1863,   80, 3,
1864,   80, 3,
1865,   80, -1.72,
1866,   80, -2.9,
1867,   80, -2.9,
1868,   80, -2.9,
1869,   80, -2.79,
1870,   80, -1.8,
1871,   80, -1.8,
1872,   80, -1.8,
1873,   80, -1.8,
1874,   80, -1.8,
1875,   80, 2.2,
1876,   80, 2.2,
1877,   80, 2.2,
1878,   80, 1.24,
1879,   80, -1,
1880,   80, -1,
1881,   80, -1,
1882,   80, -1,
1883,   80, 0.08,
1884,   80, 0.2,
1885,   80, 0.2,
1886,   80, 0.2,
1887,   80, 0.28,
1888,   80, 1,
1889,   80, 0.4,
1890,   80, -1,
1891,   80, -1,
1892,   80, -1.64,
1893,   80, -1.8,
1894,   80, -1.2,
1895,   80, -0.3,
1896,   80, -0.8,
1897,   80, -1.3,
1898,   80, -0.58,
1899,   80, -0.4,
1900,   80, -0.25,
1901,   80, -0.25,
1902,   80, -0.25,
1903,   80, -0.18,
1904,   80, 0.772,
1905,   80, 0.94,
1906,   80, -1.805,
1907,   80, -2.11,
1908,   80, -2.11,
1909,   80, -1.155,
1910,   80, -0.02,
1911,   80, -0.02,
1912,   80, 0.282,
1913,   80, 3,
1914,   80, 3,
1915,   80, 3,
1916,   80, 3,
1917,   80, 3,
1918,   80, 3,
1919,   80, 1.92,
1920,   80, 0.3,
1921,   80, 0.82,
1922,   80, 1.6,
1923,   80, 1.6,
1924,   80, 1.6,
1925,   80, 1.6,
1926,   80, 1.6,
1927,   80, 1.6,
1928,   80, 1.6,
1929,   80, 1.6,
1930,   80, 1.76,
1931,   80, 1.8,
1932,   80, 2.22,
1933,   80, 2.5,
```

```
1934,   80, 0.96,
1935,   80, 0.3,
1936,   80, 0.985,
1937,   80, 1.67,
1938,   80, 1.67,
1939,   80, 1.67,
1940,   80, 1.67,
1941,   80, 0.968,
1942,   80, 0.5,
1943,   80, 0.5,
1944,   80, -0.22,
1945,   80, -0.4,
1946,   80, -0.4,
1947,   80, -0.24,
1948,   80, 0.4,
1949,   80, 0.4,
1950,   80, 0.4,
1951,   80, 0.533,
1952,   80, 0.59,
1953,   80, -1.573,
1954,   80, -2.5,
1955,   80, -2.5,
1956,   80, -2.23,
1957,   80, -1.6,
1958,   80, -1.6,
1959,   80, -1.6,
1960,   80, -1.6,
1961,   80, -1.6,
1962,   80, 2.09,
1963,   80, 2.5,
1964,   80, 2.5,
1965,   80, 2.5,
1966,   80, 2.5,
1967,   80, 2.5,
1968,   80, 2.5,
1969,   80, 2.5,
1970,   80, 2.5,
1971,   80, 2.5,
1972,   80, 2.5,
1973,   80, 2.5,
1974,   80, 1,
1975,   80, -2.5,
1976,   80, -2.5,
1977,   80, -2.5,
1978,   80, -2.5,
1979,   80, 1.356,
1980,   80, 2.32,
1981,   80, 2.32,
1982,   80, -0.674,
1983,   80, -2.67,
1984,   80, -2.67,
1985,   80, -0.339,
1986,   80, 0.66,
1987,   80, 0.66,
1988,   80, 0.66,
1989,   80, 0.66,
1990,   80, -1.228,
1991,   80, -1.7,
1992,   80, -1.7,
1993,   80, -1.7,
1994,   80, -1.7,
1995,   80, -1.7,
1996,   80, -0.3,
1997,   80, 1.1,
1998,   80, 1.1,
1999,   80, 1.1,
2000,   80, 1.1,
2001,   80, 1.43,
2002,   80, 1.43,
2003,   80, 1.43,
2004,   80, 0.302,
2005,   80, 0.006,
2006,   80, 1.83,
2007,   80, 1.83,
2008,   80, 1.83,
2009,   80, 1.83,
2010,   80, 1.83,
2011,   80, 1.83,
2012,   80, 1.83,
2013,   80, 1.83,
2014,   80, 1.302,
2015,   80, 0.51,
2016,   80, 0.51,
2017,   80, 0.51,
2018,   80, 0.17,
2019,   80, -0.34,
2020,   80, -0.34,
2021,   80, -0.34,
```

```
2022,   80, -0.541,
2023,   80, -2.35,
2024,   80, -2.35,
2025,   80, -2.35,
2026,   80, -2.35,
2027,   80, -0.21,
2028,   80, 3,
2029,   80, 3,
2030,   80, 3,
2031,   80, 0.76,
2032,   80, 0.2,
2033,   80, 0.2,
2034,   80, 0.2,
2035,   80, 0.2,
2036,   80, 0.2,
2037,   80, -0.529,
2038,   80, -2.23,
2039,   80, -2.23,
2040,   80, -2.23,
2041,   80, -2.23,
2042,   80, -2.23,
2043,   80, -2.23,
2044,   80, -2.23,
2045,   80, -2.23,
2046,   80, -2.23,
2047,   80, -2.23,
2048,   80, -2.23,
2049,   80, -2.23,
2050,   80, -1.235,
2051,   80, -0.24,
2052,   80, -0.24,
2053,   80, -1.42,
2054,   80, -2.6,
2055,   80, -2.6,
2056,   80, -2.6,
2057,   80, -2.6,
2058,   80, -2.6,
2059,   80, -2.6,
2060,   80, -2.6,
2061,   80, -0.8,
2062,   80, 0.4,
2063,   80, 0.4,
2064,   80, 0.4,
2065,   80, 0.4,
2066,   80, 0.12,
2067,   80, 0,
2068,   80, 0,
2069,   80, 0,
2070,   80, -0.06,
2071,   80, -0.2,
2072,   80, -0.2,
2073,   80, -0.2,
2074,   80, -0.036,
2075,   80, 0.21,
2076,   80, 0.21,
2077,   80, 0.21,
2078,   80, 0.087,
2079,   80, -0.2,
2080,   80, -0.2,
2081,   80, -0.2,
2082,   80, -0.2,
2083,   80, -0.2,
2084,   80, -0.2,
2085,   80, 0.51,
2086,   80, 0.51,
2087,   80, 0.51,
2088,   80, 0.51,
2089,   80, -0.3,
2090,   80, -0.3,
2091,   80, -0.3,
2092,   80, -0.3,
2093,   80, -0.3,
2094,   80, -0.3,
2095,   80, 1.27,
2096,   80, 2.84,
2097,   80, 2.84,
2098,   80, 2.84,
2099,   80, 2.84,
2100,   80, 2.84,
2101,   80, 2.84,
2102,   80, -0.576,
2103,   80, -1.43,
2104,   80, -1.43,
2105,   80, -1.43,
2106,   80, -1.43,
2107,   80, -0.126,
2108,   80, 1.83,
2109,   80, 1.83,
```

```
2110,   80, 1.83,
2111,   80, 1.83,
2112,   80, 1.83,
2113,   80, -1.068,
2114,   80, -3,
2115,   80, -3,
2116,   80, -3,
2117,   80, -3,
2118,   80, -3,
2119,   80, -0.66,
2120,   80, -0.66,
2121,   80, -0.66,
2122,   80, -0.66,
2123,   80, -0.66,
2124,   80, -0.136,
2125,   80, 0.65,
2126,   80, 0.65,
2127,   80, 0.65,
2128,   80, 0.65,
2129,   80, 0.65,
2130,   80, 0.65,
2131,   80, 0.65,
2132,   80, 0.317,
2133,   80, -0.46,
2134,   80, -0.46,
2135,   80, -0.46,
2136,   80, -0.46,
2137,   80, 0.394,
2138,   80, 0.76,
2139,   80, 0.76,
2140,   80, 0.76,
2141,   80, 0.76,
2142,   80, 0.76,
2143,   80, 0.76,
2144,   80, 0.76,
2145,   80, 0.493,
2146,   80, -0.13,
2147,   80, -0.13,
2148,   80, -0.13,
2149,   80, -0.13,
2150,   80, -0.13,
2151,   80, -0.13,
2152,   80, -0.13,
2153,   80, 0.683,
2154,   80, 2.58,
2155,   80, 2.58,
2156,   80, 2.58,
2157,   80, 2.58,
2158,   80, 2.58,
2159,   80, 2.58,
2160,   80, 2.58,
2161,   80, 2.58,
2162,   80, 2.58,
2163,   80, 2.58,
2164,   80, 2.58,
2165,   80, 2.58,
2166,   80, 0.812,
2167,   80, 0.37,
2168,   80, 0.37,
2169,   80, 0.37,
2170,   80, 0.37,
2171,   80, -0.72,
2172,   80, -0.72,
2173,   80, -0.72,
2174,   80, 0.078,
2175,   80, 0.42,
2176,   80, 0.42,
2177,   80, 0.42,
2178,   80, 0.42,
2179,   80, 0.42,
2180,   80, 0.42,
2181,   80, 0.42,
2182,   80, 0.828,
2183,   80, 1.1,
2184,   80, 1.1,
2185,   80, 1.1,
2186,   80, 1.1,
2187,   80, 0.92,
2188,   80, -0.7,
2189,   80, -0.7,
2190,   80, -0.7,
2191,   80, -0.7,
2192,   80, 0.604,
2193,   80, 0.93,
2194,   80, 0.93,
2195,   80, 0.423,
2196,   80, -0.76,
2197,   80, -0.76,
```

```
2198,   80, -0.76,
2199,   80, -1.732,
2200,   80, -1.84,
2201,   80, -1.84,
2202,   80, -1.84,
2203,   80, -1.84,
2204,   80, 0.19,
2205,   80, 0.19,
2206,   80, 0.19,
2207,   80, 0.19,
2208,   80, 0.19,
2209,   80, 0.19,
2210,   80, 0.19,
2211,   80, 0.19,
2212,   80, 2.719,
2213,   80, 3,
2214,   80, 3,
2215,   80, 2.37,
2216,   80, 1.74,
2217,   80, 1.74,
2218,   80, 1.74,
2219,   80, 1.74,
2220,   80, 2.059,
2221,   80, 4.93,
2222,   80, 4.93,
2223,   80, 4.93,
2224,   80, 4.93,
2225,   80, 4.93,
2226,   80, 1.714,
2227,   80, -0.43,
2228,   80, -0.43,
2229,   80, -0.43,
2230,   80, -0.43,
2231,   80, -0.43,
2232,   80, -0.43,
2233,   80, -0.43,
2234,   80, -1.686,
2235,   80, -2,
2236,   80, -2,
2237,   80, -2,
2238,   80, -2,
2239,   80, -2,
2240,   80, -2,
2241,   80, -2,
2242,   80, -2,
2243,   80, -2.5,
2244,   80, -4.5,
2245,   80, -4.5,
2246,   80, -4.5,
2247,   80, -4.5,
2248,   80, -4.5,
2249,   80, -4.5,
2250,   80, -2.8,
2251,   80, 4,
2252,   80, 4,
2253,   80, 4,
2254,   80, 4,
2255,   80, 4,
2256,   80, 4,
2257,   80, 4,
2258,   80, 4,
2259,   80, 4,
2260,   80, 4,
2261,   80, 1.09,
2262,   80, -0.85,
2263,   80, -0.85,
2264,   80, -0.85,
2265,   80, -1.555,
2266,   80, -3.2,
2267,   80, -3.2,
2268,   80, -3.2,
2269,   80, -3.2,
2270,   80, -1.048,
2271,   80, 2.18,
2272,   80, 2.18,
2273,   80, 2.18,
2274,   80, 2.18,
2275,   80, 0.398,
2276,   80, 0.2,
2277,   80, 0.2,
2278,   80, 0.2,
2279,   80, 0.2,
2280,   80, 0.2,
2281,   80, 0.2,
2282,   80, 0.2,
2283,   80, 0.2,
2284,   80, 0.2,
2285,   80, -0.385,
```

```
2286,   80, -0.45,
2287,   80, -0.45,
2288,   80, -0.45,
2289,   80, -0.45,
2290,   80, -0.45,
2291,   80, -0.45,
2292,   80, -0.45,
2293,   80, -0.45,
2294,   80, -0.45,
2295,   80, -0.222,
2296,   80, -0.07,
2297,   80, -0.07,
2298,   80, -0.07,
2299,   80, -0.07,
2300,   80, -0.07,
2301,   80, -0.07,
2302,   80, -0.07,
2303,   80, -0.083,
2304,   80, -0.2,
2305,   80, -0.2,
2306,   80, -0.2,
2307,   80, -0.2,
2308,   80, -0.2,
2309,   80, -0.2,
2310,   80, 0.02,
2311,   80, 2,
2312,   80, 2,
2313,   80, 2,
2314,   80, 2,
2315,   80, 2,
2316,   80, 2,
2317,   80, 2,
2318,   80, 2,
2319,   80, 2,
2320,   80, -1.24,
2321,   80, -1.6,
2322,   80, -1.6,
2323,   80, -1.24,
2324,   80, 0.2,
2325,   80, 0.2,
2326,   80, 0.2,
2327,   80, 0.2,
2328,   80, 0.2,
2329,   80, -0.25,
2330,   80, -0.3,
2331,   80, -0.3,
2332,   80, -0.3,
2333,   80, -0.22,
2334,   80, -0.2,
2335,   80, -0.2,
2336,   80, -0.2,
2337,   80, -0.2,
2338,   80, -0.14,
2339,   80, 0.1,
2340,   80, 0.1,
2341,   80, 0.1,
2342,   80, 0.1,
2343,   80, 0.1,
2344,   80, 0.1,
2345,   80, 0.1,
2346,   80, 0.1,
2347,   80, 0.1,
2348,   80, 0.1,
2349,   80, 2.5,
2350,   80, 2.5,
2351,   80, 2.5,
2352,   80, 2.5,
2353,   80, 2.5,
2354,   80, -1.406,
2355,   80, -1.84,
2356,   80, -1.84,
2357,   80, -1.84,
2358,   80, -1.84,
2359,   80, -1.84,
2360,   80, -1.84,
2361,   80, -1.84,
2362,   80, -1.84,
2363,   80, -0.208,
2364,   80, 0.88,
2365,   80, 0.88,
2366,   80, 0.88,
2367,   80, 0.88,
2368,   80, 0.88,
2369,   80, 0.88,
2370,   80, 0.88,
2371,   80, 0.88,
2372,   80, -0.366,
2373,   80, -0.9,
```

```
2374,   80, -0.9,
2375,   80, -0.9,
2376,   80, -0.102,
2377,   80, 0.43,
2378,   80, 0.43,
2379,   80, 0.43,
2380,   80, 0.43,
2381,   80, 0.43,
2382,   80, 0.045,
2383,   80, -0.34,
2384,   80, -0.34,
2385,   80, -0.34,
2386,   80, -0.34,
2387,   80, -0.34,
2388,   80, -0.34,
2389,   80, -0.34,
2390,   80, -0.34,
2391,   80, -0.293,
2392,   80, 0.13,
2393,   80, 0.13,
2394,   80, 0.618,
2395,   80, 1.35,
2396,   80, 1.35,
2397,   80, 1.35,
2398,   80, 1.35,
2399,   80, 1.35,
2400,   80, 1.35,
2401,   80, 1.35,
2402,   80, 1.35,
2403,   80, 1.35,
2404,   80, 1.35,
2405,   80, 1.35,
2406,   80, 1.105,
2407,   80, 0.86,
2408,   80, 0.86,
2409,   80, 0.86,
2410,   80, 0.86,
2411,   80, 0.541,
2412,   80, -2.33,
2413,   80, -2.33,
2414,   80, -2.33,
2415,   80, -2.33,
2416,   80, -2.33,
2417,   80, -2.33,
2418,   80, -2.33,
2419,   80, -2.33,
2420,   80, -2.33,
2421,   80, -2.33,
2422,   80, -1.23,
2423,   80, -0.13,
2424,   80, -0.13,
2425,   80, -0.13,
2426,   80, -0.13,
2427,   80, -0.13,
2428,   80, -0.13,
2429,   80, -0.08,
2430,   80, 0.37,
2431,   80, 0.37,
2432,   80, 0.37,
2433,   80, 0.37,
2434,   80, 0.37,
2435,   80, 0.37,
2436,   80, 0.37,
2437,   80, 0.37,
2438,   80, 0.37,
2439,   80, 0.37,
2440,   80, 0.37,
2441,   80, 0.37,
2442,   80, 0.37,
2443,   80, 0.37,
2444,   80, 0.346,
2445,   80, 0.29,
2446,   80, 0.29,
2447,   80, 0.29,
2448,   80, 0.29,
2449,   80, 0.297,
2450,   80, 0.36,
2451,   80, 0.36,
2452,   80, 0.36,
2453,   80, 0.36,
2454,   80, 0.257,
2455,   80, -0.67,
2456,   80, -0.67,
2457,   80, -0.67,
2458,   80, -0.403,
2459,   80, 2,
2460,   80, 2,
2461,   80, 2,
```

```
2462,   80, 0.692,
2463,   80, -2.36,
2464,   80, -2.36,
2465,   80, -2.36,
2466,   80, -2.36,
2467,   80, -2.36,
2468,   80, -2.36,
2469,   80, -2.36,
2470,   80, -2.36,
2471,   80, -0.962,
2472,   80, 2.3,
2473,   80, 2.3,
2474,   80, 2.3,
2475,   80, 2.3,
2476,   80, 2.3,
2477,   80, 2.3,
2478,   80, 2.3,
2479,   80, 2.3,
2480,   80, -1.772,
2481,   80, -2.79,
2482,   80, -2.79,
2483,   80, -2.79,
2484,   80, -2.79,
2485,   80, -2.232,
2486,   80, 0,
2487,   80, 0,
2488,   80, 0,
2489,   80, 0,
2490,   80, 0.228,
2491,   80, 0.76,
2492,   80, 0.76,
2493,   80, 0.76,
2494,   80, 0.76,
2495,   80, 0.76,
2496,   80, 0.76,
2497,   80, 0.76,
2498,   80, 0.76,
2499,   80, -0.248,
2500,   80, -0.5,
2501,   80, -0.5,
2502,   80, -0.5,
2503,   80, -0.15,
2504,   80, 0,
2505,   80, 0,
2506,   80, 0,
2507,   80, 1.8,
2508,   80, 2,
2509,   80, 2,
2510,   80, 1.04,
2511,   80, 0.4,
2512,   80, 0.4,
2513,   80, 0.4,
2514,   80, 0.4,
2515,   80, 0.4,
2516,   80, 0.4,
2517,   80, 0.4,
2518,   80, -0.05,
2519,   80, -0.5,
2520,   80, -0.5,
2521,   80, -0.5,
2522,   80, -0.5,
2523,   80, -0.5,
2524,   80, -0.5,
2525,   80, -0.5,
2526,   80, 2.335,
2527,   80, 2.65,
2528,   80, 2.65,
2529,   80, 2.65,
2530,   80, 2.65,
2531,   80, 2.65,
2532,   80, 2.65,
2533,   80, 2.65,
2534,   80, 2.65,
2535,   80, 2.65,
2536,   80, -0.5,
2537,   80, -2.6,
2538,   80, -2.6,
2539,   80, -2.6,
2540,   80, -2.6,
2541,   80, -2.6,
2542,   80, -2.6,
2543,   80, -2.6,
2544,   80, -2.6,
2545,   80, -2.6,
2546,   80, -0.88,
2547,   80, -0.45,
2548,   80, -0.45,
2549,   80, -0.45,
```

```
2550,   80, 0.192,
2551,   80, 2.76,
2552,   80, 2.76,
2553,   80, 2.76,
2554,   80, 2.76,
2555,   80, 2.76,
2556,   80, 2.76,
2557,   80, 2.76,
2558,   80, 2.76,
2559,   80, 2.76,
2560,   80, 2.76,
2561,   80, 2.76,
2562,   80, 2.76,
2563,   80, 2.76,
2564,   80, -0.76,
2565,   80, -0.76,
2566,   80, -0.76,
2567,   80, -0.76,
2568,   80, -0.76,
2569,   80, -0.76,
2570,   80, -0.76,
2571,   80, -0.76,
2572,   80, -0.76,
2573,   80, 0.208,
2574,   80, 0.45,
2575,   80, 0.45,
2576,   80, 0.45,
2577,   80, -0.6,
2578,   80, -1.3,
2579,   80, -1.3,
2580,   80, -1.3,
2581,   80, -0.148,
2582,   80, 0.62,
2583,   80, 0.62,
2584,   80, 0.62,
2585,   80, 0.03,
2586,   80, -2.33,
2587,   80, -2.33,
2588,   80, -2.33,
2589,   80, -2.33,
2590,   80, -1.02,
2591,   80, 0.29,
2592,   80, 0.29,
2593,   80, 0.29,
2594,   80, 0.29,
2595,   80, 0.29,
2596,   80, -1.222,
2597,   80, -1.6,
2598,   80, -1.6,
2599,   80, -1.6,
2600,   80, -1.6,
2601,   80, -1.6,
2602,   80, -1.251,
2603,   80, 1.89,
2604,   80, 1.89,
2605,   80, 1.89,
2606,   80, 1.89,
2607,   80, 0.217,
2608,   80, -0.5,
2609,   80, -0.5,
2610,   80, -0.5,
2611,   80, -0.5,
2612,   80, -0.5,
2613,   80, -0.5,
2614,   80, 0.154,
2615,   80, 1.68,
2616,   80, 1.68,
2617,   80, 1.68,
2618,   80, 1.68,
2619,   80, 1.68,
2620,   80, 1.68,
2621,   80, 1.68,
2622,   80, 1.68,
2623,   80, 0.632,
2624,   80, 0.37,
2625,   80, 0.37,
2626,   80, 0.37,
2627,   80, 0.37,
2628,   80, 0.37,
2629,   80, 0.37,
2630,   80, 0.37,
2631,   80, 0.37,
2632,   80, 2.25,
2633,   80, 2.72,
2634,   80, 2.72,
2635,   80, 2.72,
2636,   80, 2.72,
2637,   80, 0.866,
```

```
2638,   80, 0.66,
2639,   80, 0.66,
2640,   80, 0.66,
2641,   80, 0.66,
2642,   80, -0.012,
2643,   80, -0.3,
2644,   80, -0.3,
2645,   80, -0.3,
2646,   80, -0.3,
2647,   80, -0.3,
2648,   80, -0.3,
2649,   80, -0.3,
2650,   80, -0.3,
2651,   80, -0.3,
2652,   80, -0.3,
2653,   80, -0.3,
2654,   80, -0.3,
2655,   80, 0.18,
2656,   80, 1.3,
2657,   80, 1.3,
2658,   80, 1.3,
2659,   80, 1.3,
2660,   80, 1.3,
2661,   80, -0.149,
2662,   80, -0.77,
2663,   80, -0.77,
2664,   80, -0.77,
2665,   80, -0.77,
2666,   80, -0.77,
2667,   80, -0.77,
2668,   80, -0.77,
2669,   80, -0.77,
2670,   80, -0.77,
2671,   80, -0.77,
2672,   80, -0.77,
2673,   80, -0.77,
2674,   80, 3,
2675,   80, 3,
2676,   80, 3,
2677,   80, 3,
2678,   80, 3,
2679,   80, 3,
2680,   80, 2.63,
2681,   80, 2.26,
2682,   80, 2.26,
2683,   80, 2.26,
2684,   80, 2.26,
2685,   80, 2.26,
2686,   80, 2.26,
2687,   80, 2.26,
2688,   80, 2.26,
2689,   80, 2.26,
2690,   80, 2.26,
2691,   80, 2.26,
2692,   80, 2.26,
2693,   80, 2.26,
2694,   80, 2.26,
2695,   80, 2.26,
2696,   80, 2.26,
2697,   80, 2.917,
2698,   80, 2.99,
2699,   80, 2.99,
2700,   80, 2.99,
2701,   80, 2.99,
2702,   80, 2.644,
2703,   80, 1.26,
2704,   80, 1.26,
2705,   80, 1.26,
2706,   80, 1.26,
2707,   80, 1.26,
2708,   80, 1.26,
2709,   80, 1.26,
2710,   80, 1.26,
2711,   80, 1.26,
2712,   80, 1.26,
2713,   80, 1.26,
2714,   80, 1.855,
2715,   80, 2.45,
2716,   80, 2.45,
2717,   80, 2.45,
2718,   80, 2.09,
2719,   80, 2.05,
2720,   80, 2.05,
2721,   80, 2.05,
2722,   80, 2.05,
2723,   80, 2.05,
2724,   80, 1.873,
2725,   80, 1.46,
```

```
2726,   80, 1.46,
2727,   80, 1.46,
2728,   80, 1.46,
2729,   80, 1.218,
2730,   80, 0.25,
2731,   80, 0.25,
2732,   80, 0.25,
2733,   80, 0.325,
2734,   80, 1,
2735,   80, 1,
2736,   80, 1,
2737,   80, 0.18,
2738,   80, -0.64,
2739,   80, -0.64,
2740,   80, -0.64,
2741,   80, -0.64,
2742,   80, -0.64,
2743,   80, -1.347,
2744,   80, -1.65,
2745,   80, -1.65,
2746,   80, 0.51,
2747,   80, 0.75,
2748,   80, 0.75,
2749,   80, 0.75,
2750,   80, 0.75,
2751,   80, 0.75,
2752,   80, 0.126,
2753,   80, -2.37,
2754,   80, -2.37,
2755,   80, -2.37,
2756,   80, -2.696,
2757,   80, -4,
2758,   80, -4,
2759,   80, -4,
2760,   80, -4,
2761,   80, -4,
2762,   80, -4,
2763,   80, -4,
2764,   80, 0.77,
2765,   80, 0.77,
2766,   80, 0.77,
2767,   80, 0.77,
2768,   80, 0.77,
2769,   80, 0.77,
2770,   80, 0.77,
2771,   80, 0.77,
2772,   80, 0.77,
2773,   80, -2.515,
2774,   80, -2.88,
2775,   80, -2.88,
2776,   80, -2.88,
2777,   80, -2.88,
2778,   80, -2.88,
2779,   80, -1.12,
2780,   80, -1.12,
2781,   80, -1.12,
2782,   80, -1.12,
2783,   80, -1.12,
2784,   80, -1.12,
2785,   80, -3,
2786,   80, -3,
2787,   80, -3,
2788,   80, -3,
2789,   80, -3,
2790,   80, -1.04,
2791,   80, -0.55,
2792,   80, -0.55,
2793,   80, -1.37,
2794,   80, -2.6,
2795,   80, -2.6,
2796,   80, -2.6,
2797,   80, -2.6,
2798,   80, -2.6,
2799,   80, 0.048,
2800,   80, 0.71,
2801,   80, 0.71,
2802,   80, 0.563,
2803,   80, 0.5,
2804,   80, 0.5,
2805,   80, 0.5,
2806,   80, -1.244,
2807,   80, -1.68,
2808,   80, -1.68,
2809,   80, -1.548,
2810,   80, -1.35,
2811,   80, -1.35,
2812,   80, -1.35,
2813,   80, -1.35,
```

```
2814,   80, -1.35,
2815,   80, -1.754,
2816,   80, -2.36,
2817,   80, -2.36,
2818,   80, -2.36,
2819,   80, -2.36,
2820,   80, -3.164,
2821,   80, -5.04,
2822,   80, -5.04,
2823,   80, -5.04,
2824,   80, -5.04,
2825,   80, -4.74,
2826,   80, -2.04,
2827,   80, -2.04,
2828,   80, -2.04,
2829,   80, -1.724,
2830,   80, -1.25,
2831,   80, -1.25,
2832,   80, -1.233,
2833,   80, -1.08,
2834,   80, -1.53,
2835,   80, -0.012,
2836,   80, 2.94,
2837,   80, 1.926,
2838,   80, -0.44,
2839,   80, -0.44,
2840,   80, -2.652,
2841,   80, -3.6,
2842,   80, -3.07,
2843,   80, -2.54,
2844,   80, -2.108,
2845,   80, 2.797,
2846,   80, 2.144,
2847,   80, -2.6,
2848,   80, -2.6,
2849,   80, 0.757,
2850,   80, 1.13,
2851,   80, 1.515,
2852,   80, 1.68,
2853,   80, 1.68,
2854,   80, 1.68,
2855,   80, 1.68,
2856,   80, 1.68,
2857,   80, 1.68,
2858,   80, 1.68,
2859,   80, 3.546,
2860,   80, 4.79,
2861,   80, 4.011,
2862,   80, -3,
2863,   80, -3,
2864,   80, -3,
2865,   80, -2.97,
2866,   80, -2.7,
2867,   80, -2.7,
2868,   80, -2.7,
2869,   80, -2.7,
2870,   80, 1.004,
2871,   80, 1.93,
2872,   80, 1.93,
2873,   80, 1.93,
2874,   80, 1.93,
2875,   80, 1.93,
2876,   80, -2.334,
2877,   80, -3.4,
2878,   80, -2.585,
2879,   80, 4.75,
2880,   80, 4.75,
2881,   80, 4.75,
2882,   80, 2.092,
2883,   80, 0.32,
2884,   80, 0.32,
2885,   80, -0.778,
2886,   80, -0.9,
2887,   80, -0.9,
2888,   80, -0.9,
2889,   80, -0.9,
2890,   80, -1.596,
2891,   80, -3.22,
2892,   80, -3.22,
2893,   80, -3.22,
2894,   80, -3.22,
2895,   80, -3.22,
2896,   80, -1.492,
2897,   80, -1.3,
2898,   80, -1.3,
2899,   80, -1.3,
2900,   80, -1.3,
2901,   80, -1.3,
```

```
2902,   80, -2.6,
2903,   80, -2.6,
2904,   80, -0.76,
2905,   80, -0.3,
2906,   80, -0.3,
2907,   80, -0.3,
2908,   80, -0.155,
2909,   80, -0.01,
2910,   80, -0.01,
2911,   80, -0.01,
2912,   80, -0.01,
2913,   80, -0.126,
2914,   80, -0.3,
2915,   80, -0.3,
2916,   80, 1.239,
2917,   80, 1.41,
2918,   80, 1.41,
2919,   80, 1.41,
2920,   80, 1.41,
2921,   80, 1.135,
2922,   80, 0.86,
2923,   80, 0.86,
2924,   80, 0.86,
2925,   80, 0.86,
2926,   80, 0.316,
2927,   80, -1.86,
2928,   80, -1.86,
2929,   80, -1.86,
2930,   80, -1.86,
2931,   80, -1.86,
2932,   80, -1.455,
2933,   80, -1.41,
2934,   80, -1.41,
2935,   80, -1.41,
2936,   80, -1.41,
2937,   80, -1.68,
2938,   80, -1.95,
2939,   80, -1.95,
2940,   80, 0.122,
2941,   80, 0.64,
2942,   80, 0.54,
2943,   80, -0.36,
2944,   80, -0.36,
2945,   80, -0.36,
2946,   80, -0.36,
2947,   80, -0.36,
2948,   80, 0.249,
2949,   80, 0.51,
2950,   80, 0.51,
2951,   80, 0.51,
2952,   80, 0.51,
2953,   80, -0.456,
2954,   80, -1.1,
2955,   80, -1.1,
2956,   80, -0.868,
2957,   80, 0.06,
2958,   80, 0.06,
2959,   80, 0.06,
2960,   80, 0.06,
2961,   80, 0.06,
2962,   80, -0.09,
2963,   80, -0.24,
2964,   80, -0.24,
2965,   80, -0.24,
2966,   80, -0.24,
2967,   80, -0.24,
2968,   80, 0.036,
2969,   80, 0.22,
2970,   80, 0.22,
2971,   80, 0.745,
2972,   80, 1.97,
2973,   80, 1.97,
2974,   80, 1.97,
2975,   80, 1.97,
2976,   80, 1.97,
2977,   80, 1.97,
2978,   80, 1.97,
2979,   80, -0.1,
2980,   80, -1.131,
2981,   80, -1.22,
2982,   80, -1.22,
2983,   80, 1.006,
2984,   80, 1.96,
2985,   80, 1.96,
2986,   80, 0.472,
2987,   80, -0.52,
2988,   80, -0.52,
2989,   80, -0.52,
```

```
2990,   80, -0.52,
2991,   80, -0.52,
2992,   80, -0.52,
2993,   80, 0.4,
2994,   80, 0.4,
2995,   80, 0.4,
2996,   80, 0.4,
2997,   80, 0.4,
2998,   80, 0.4,
2999,   80, 0.4,
3000,   80, 1.582,
3001,   80, 2.37,
3002,   80, 2.37,
3003,   80, 2.37,
3004,   80, 2.37,
3005,   80, 2.37,
3006,   80, 2.37,
3007,   80, 2.37,
3008,   80, 1.998,
3009,   80, 0.51,
3010,   80, 0.51,
3011,   80, 0.51,
3012,   80, 0.221,
3013,   80, -2.38,
3014,   80, -2.38,
3015,   80, -2.38,
3016,   80, -0.292,
3017,   80, 0.23,
3018,   80, 0.23,
3019,   80, 0.288,
3020,   80, 0.52,
3021,   80, 0.52,
3022,   80, 0.52,
3023,   80, 1.6,
3024,   80, 1.87,
3025,   80, 0.586,
3026,   80, -2.41,
3027,   80, -2.41,
3028,   80, -2.41,
3029,   80, -2.41,
3030,   80, -2.41,
3031,   80, -0.996,
3032,   80, -1.122,
3033,   80, -2.83,
3034,   80, -2.83,
3035,   80, -2.83,
3036,   80, -2.83,
3037,   80, -2.83,
3038,   80, -2.83,
3039,   80, -2.83,
3040,   80, -2.83,
3041,   80, -2.247,
3042,   80, 3,
3043,   80, 3,
3044,   80, 3,
3045,   80, 3,
3046,   80, 3,
3047,   80, 3,
3048,   80, -0.576,
3049,   80, -2.96,
3050,   80, -2.96,
3051,   80, -2.96,
3052,   80, -2.96,
3053,   80, -2.96,
3054,   80, -1.786,
3055,   80, 2.91,
3056,   80, 2.91,
3057,   80, 2.91,
3058,   80, 2.91,
3059,   80, 2.91,
3060,   80, -0.83,
3061,   80, -0.83,
3062,   80, -0.83,
3063,   80, -0.83,
3064,   80, -0.768,
3065,   80, -0.52,
3066,   80, -0.52,
3067,   80, -0.52,
3068,   80, -0.52,
3069,   80, -0.52,
3070,   80, 2.639,
3071,   80, 2.99,
3072,   80, 2.99,
3073,   80, 2.99,
3074,   80, 2.99,
3075,   80, 2.99,
3076,   80, -1.162,
3077,   80, -2.2,
```

```
3078,   80, -2.2,
3079,   80, -2.2,
3080,   80, -2.2,
3081,   80, -2.2,
3082,   80, -2.2,
3083,   80, -2.2,
3084,   80, -2.2,
3085,   80, -2.2,
3086,   80, 1.08,
3087,   80, 1.08,
3088,   80, 1.08,
3089,   80, -0.712,
3090,   80, 1.78,
3091,   80, 4,
3092,   80, 4,
3093,   80, 4,
3094,   80, -1.4,
3095,   80, -5,
3096,   80, -5,
3097,   80, -3.135,
3098,   80, -1.27,
3099,   80, -1.27,
3100,   80, -0.494,
3101,   80, -0.3,
3102,   80, 0.49,
3103,   80, 0.49,
3104,   80, 0.49,
3105,   80, 0.49,
3106,   80, 0.49,
3107,   80, 0.669,
3108,   80, 2.28,
3109,   80, 2.28,
3110,   80, 2.28,
3111,   80, 2.28,
3112,   80, 2.28,
3113,   80, 2.28,
3114,   80, 2.28,
3115,   80, 0.228,
3116,   80, 0,
3117,   80, 0,
3118,   80, 0,
3119,   80, 0};
```