```cpp
/*********************************************************************************
PROGRAM       : lunar_lander.cpp
PROGRAM PURPOSE: The is a small game named Lunar Lander, which is based on a variety of similarly themed
                 games inspired by the Apollo 11 space mission of July, 1969 in which US astronauts first
                 landed on the Moon. After running the program, it'll engage the user in a session that plays
                 the game, updates the values of height and velocity at each turn and stops when the spacecraft
                 reaches the surface (aka touchdown). At this point, a final analysis is printed.
Coder         : Mosfiqur Rahman (mr986@drexel.edu)
Last Modified : 11th March, 2016.
*********************************************************************************/

#include <iostream> // for having input and output
#include <cmath>   //  for advanced mathematical cal calculation
#include <string>  // for using string properties
#include <fstream> // for getting input from a file and setting output to a file
#include <algorithm> // for using in-built algorithms

using namespace std;  //for using standard library namespace

void reportStatus(ostream &os, double time, double height, double velocity, double fuel, string name);
// Prototyping function reportStatus, which will output informations related with the time, height, speed, and
fuel repeatedly

void updateStatus(double &velocity, double burnAmount, double &fuelRemaining, double &height);
// Prototyping function updateStatus, which  uses the current values of velocity, burnAmount, fuelRemaining and
//height to compute the values one second later

void introduction(istream &is, ostream &os, string target,string replacement);
// Prototyping function introduction, which will create a set of instructions replacing all instances of some
string target
//from the input stream with the string specified by the replacement parameter for the output stream

void touchdown(double &elapsedTime, double &velocity, double&burnAmount, double &height);
//Prototyping the function touchdown, which will do some corrections at the point of touchdown in order to
//get accurate final values of height and velocity

void finalAnalysis(ostream &os, double velocity);
//Prototyping the function finalAnalysis, which will report the damage based on the final velocity

//The following function is the main function
int main()
{

double height = 1000;
// Declares a double type variable named height, and initializes the value to 1000 feet

double velocity = 50;
// Declares a double type variable named velocity, and initializes the value to 50feet/sec

double time = 0;
//Declares a double type variable named time, and initializes the value to 0sec

double fuel = 150;
//Declares a double type variable named fuel, and initializes the value to 150 units

double burn_fuel;
//Declares a double type variable named burn_fuel

string name = "APOLLO";
//Declares a string type variable named name, and initializes the value to "APOLLO"

string filename;
//Declares a string type variable named filename

    string check;
    //Declares a string type variable named check
    string target = "$SPACECRAFT";
    //Declares a string type variable named target, and initializes the value to "$SPACECRAFT"
    ifstream is("input.txt");
    //Declares a ifstraem object is and sets the file name to "input.txt"
    cout << "Please, enter the file name where you would like to log you session: ";
    // Asks the user to file name where they would like to log their session
    cin >> filename;
    // Takes the input from user5 and stores the file name in a string variable name filename

    ofstream os(filename.c_str());
    //Declares a ofstraem object os and sets the file name as directed by the user


    cout << endl <<"LUNAR LANDER" << endl << "By Dave Ahl (translation from BASIC to ELEVEN by Joe Morrison)" <<
endl << endl << "Do you want instructions (y/n)? ";
    // Asks the user whether they wants to see the instruction or not
    os   << endl <<"LUNAR LANDER" << endl << "By Dave Ahl (translation from BASIC to ELEVEN by Joe Morrison)" <<
endl << endl << "Do you want instructions (y/n)? ";
    //Setting output to the output file as directed by the user
    cin  >> check;
    // Takes the response from the user
    os   << check;
    cout << endl;
    os   << endl;

    if (check == "Y" || check == "y")
        // Checks whether the value of check is equal to Y or y
       //If it's so then continues
    {
        introduction( is, os, target, name);
    // Calls the function introduction(istream &is, ostream &os, string target,string replacement), and passes
the value
    }
    cout << endl << "LUNAR LANDER" << endl << "Beginning landing procedure.........." << endl << "DIGBY wishes
you good luck !!!!!!!" << endl;
    // Optional texts
    os   << endl << "LUNAR LANDER" << endl << "Beginning landing procedure.........." << endl << "DIGBY wishes
you good luck !!!!!!!" << endl ;
    //Setting output to the output file as directed by the user


    reportStatus(os, time,  height, velocity, fuel, name);
    // Calls the function reportStatus(ostream &os, double time, double height, double velocity, double fuel,
string name) and passes the value

    time++;
    // Increments the value of time


 while( cout << "Enter fuel burn amount: " && (cin >> burn_fuel) )
    // Runs a while loop to get the amount of fuel that the user would like to burn
 {
     os << "Enter fuel burn amount: " << burn_fuel;
     if ((burn_fuel > 30) || (burn_fuel < 0) )
     // Checks whether the value of burn_fuel is greater than 30 or less than 0
    //If it's so then continues, or moves to other conditional statement.

     {
         cout << "Sorry, please input a valid positive number[Starting from 0 to 30]." << endl;
         // Alert message
         os   << "Sorry, please input a valid positive number[Starting from 0 to 30]." << endl;
         //Setting output to the output file as directed by the user
         continue;
         // continues the loop from the beginning again
     }
     else if((fuel > 0) && (burn_fuel > fuel))
```

```
        // Checks whether the value of fuel is greater than 0 or the value of burn_fuel is greater than fuel
    //If it's so then continues, or moves to other conditional statement.

    {
        burn_fuel = fuel;
        // Initializes the value of the variable burn_fuel
        updateStatus(velocity, burn_fuel, fuel, height);
        // Calls the function updateStatus(double &velocity, double burnAmount, double &fuelRemaining, double
&height) and passes the value
        reportStatus(os, time,  height, velocity, fuel, name);
        // Calls the function reportStatus(ostream &os, double time, double height, double velocity, double
fuel, string name) and passes the value


       if(fuel == 0 || height <= 0)
           // Checks whether the value of fuel is equal to 0 or the value of height is less than or equal to 0
   //If it's so then continues, or moves to other conditional statement.

    {
        burn_fuel = 0;
        //Sets the value of the burn fuel to 0

        if (fuel == 0)
        {
            cout << "**** OUT OF FUEL ****";
        // Alert message
            os   << "**** OUT OF FUEL ****";
        }
        cout << endl << endl << "***** CONTACT *****" << endl;
        os   << endl << endl << "***** CONTACT *****" << endl;
        // Alert message

        touchdown(time, velocity, burn_fuel, height);
        //Calls the function touchdown(double &elapsedTime, double &velocity, double&burnAmount, double
&height) and passes the value

        cout << "Touchdown at " << time << " seconds." << endl;
        // Shows final output for touchdown time
        cout << "Landing velocity = " << velocity << " feet/sec" << endl;
        // Shows final output for landing velocity
        cout << fuel << " units of fuel remaining." << endl << endl;
        // Shows final output for remaining fuel

        os << "Touchdown at " << time << " seconds." << endl;
        os << "Landing velocity = " << velocity << " feet/sec" << endl;
        os << fuel << " units of fuel remaining." << endl << endl;
        //Setting output to the output file as directed by the user

        finalAnalysis(os, velocity);
        //Calls the function finalAnalysis(ostream &os, double velocity) and passes the value
        break;
        // breaks the loop
    }
        time++;
        // increments the value of time
        continue;
        // continues the loop from the beginning again
    }

    else
       // If the previous 'if' and 'else if' statements fail, then the program
      // run through this statement.


    {
       updateStatus(velocity, burn_fuel, fuel, height);
```

```
        // Calls the function updateStatus(double &velocity, double burnAmount, double &fuelRemaining, double
&height) and passes the value

        reportStatus(os, time,  height, velocity, fuel, name);
        // Calls the function reportStatus(ostream &os, double time, double height, double velocity, double
fuel, string name) and passes the value


        if(fuel == 0 || height <= 0)
            // Checks whether the value of fuel is equal to 0 or the value of height is less than or equal to 0
    //If it's so then continues, or moves to other conditional statement.

     {

         burn_fuel = 0;
         //Sets the value of the burn fuel to 0
         if (fuel == 0)
         {
             cout << "**** OUT OF FUEL ****";
         // Alert message
             os   << "**** OUT OF FUEL ****";
         }
         cout << endl << endl << "***** CONTACT *****" << endl;
         os   << endl << endl << "***** CONTACT *****" << endl;
         // Alert message

         touchdown(time, velocity, burn_fuel, height);
         //Calls the function touchdown(double &elapsedTime, double &velocity, double&burnAmount, double
&height) and passes the value

         cout << "Touchdown at " << time << " seconds." << endl;
         // Shows final output for touchdown time
         cout << "Landing velocity = " << velocity << " feet/sec" << endl;
         // Shows final output for landing velocity
         cout << fuel << " units of fuel remaining." << endl << endl;
         // Shows final output for remaining fuel

         os << "Touchdown at " << time << " seconds." << endl;
         os << "Landing velocity = " << velocity << " feet/sec" << endl;
         os << fuel << " units of fuel remaining." << endl << endl;
         //Setting output to the output file as directed by the user

         finalAnalysis(os, velocity);
         //Calls the function finalAnalysis(ostream &os, double velocity) and passes the value
         break;
         // breaks the loop
     }
        time++;
        // increments the value of time
        continue;
        // continues the loop from the beginning again
     }

     os.close();
     //Closes the output object

 }

return 0;
}

/***
                This function outputs informations related with the time, height, speed, and fuel repeatedly

 @param &os       - The ofstream object
 @param time     - The elapsed time
```

```cpp
 @param height   - The height from the ground
 @param velocity - The velocity of the falling spaceship
 @param fuel     - The amount of remaining fuel
 @param name     - The name of the spaceship, i.e. APOLLO


 ***/

void reportStatus(ostream &os, double time, double height, double velocity, double fuel, string name)
// Defining function reportStatus, which will output informations related with the time, height, speed, and fuel
repeatedly
{

        os << endl << endl << "Status of your " << name << " spacecraft:" << endl;
        os << "Time : " << time << " seconds" << endl;
        os << "Height:" << height << " feet" << endl;
        os << "Speed :" << velocity << " feet/second" << endl;
        os << "Fuel : " << fuel << endl;
        //Setting output to the output file as directed by the user

        cout << endl << endl << "Status of your " << name << " spacecraft:" << endl;
        // Introductory text
        cout << "Time : " << time << " seconds" << endl;
        // Shows the elapsed time as output
        cout << "Height:" << height << " feet" << endl;
        // Shows the certain height from the ground
        cout << "Speed :" << velocity << " feet/second" << endl;
        // Shows the speed of the spaceship
        cout << "Fuel : " << fuel << endl;
        // Shows the amount of remaining fuel


}

/***
                This function uses the current values of velocity, burnAmount, fuelRemaining and
                height to compute the values one second later

 @param &height        - The height from the ground
 @param &velocity      - The velocity of the falling spaceship
 @param &fuelRemaining - The amount of remaining fuel
 @param burnAmount     - The amount of fuel that is going to be burnt


 ***/



void updateStatus(double &velocity, double burnAmount, double &fuelRemaining, double &height)
//Defining function updateStatus, which  uses the current values of velocity, burnAmount, fuelRemaining and
//height to compute the values one second later

{
    double old_velocity = velocity;
    // Declares a double type variable named old_velocity, and initializes the value to velocity
    double old_height   = height;
    // Declares a double type variable named old_height, and initializes the value to height
    double old_fuel = fuelRemaining;
    // Declares a double type variable named old_fuel, and initializes the value to fuelRemaining

    fuelRemaining = old_fuel - burnAmount;
    // Computes the value of the variable fuelRemaining
    velocity = old_velocity + 5 - burnAmount;
    // As acceleration due to gravity is set to 5
    // Computes the value of the variable velocity
    height   = old_height - (old_velocity + velocity)/2;
    // Computes the value of the variable height
    if (height <= 0)
    {
        height = 0;
    }

}


/***
                This function creates a set of instructions replacing all instances of some string target
                from the input stream with the string specified by the replacement parameter for the output
stream

 @param &is          - The ifstream object
 @param &os          - The ofstream object
 @param target       - The specified string, which is going to be replaced to
 @param replacement - The specified string, which is going to be replaced with


 ***/


void introduction(istream &is, ostream &os, string target,string replacement)
//Defining function introduction, which will create a set of instructions replacing all instances of some string
target
//from the input stream with the string specified by the replacement parameter for the output stream

{
    string line;
    //Declares a string variable named line
    string replacement_final;
    //Declares a string variable named replacement_final

    while(getline(is,line))
    //Runs awhile loop to find the targeted string from the file
    {
        int index = line.find(target);
        // Declares a integer type variable named index, and initializes the value to the starting point of the
targeted string

        while (index >= 0)
        // Runs a while loop to replace the targeted string
        {
            replacement_final = line.replace(index, target.length(), replacement);
            //replaces the targeted string
            line = replacement_final;
            //Initializes the value of of the string line to replacement_final
            index = line.find(target);
            //Initializes the value of the variable index to the starting point of the targeted string
        }
            os << line << endl;
            //Setting output to the output file as directed by the user

            cout << line << endl;
            // Shows the output of new with the new value of the string
        }

}


/***
                This function does some corrections at the point of touchdown in order to
                get accurate final values of height and velocity

 @param &height         - The height from the ground
 @param &velocity      - The velocity of the falling spaceship
 @param &elapsedTime   - The amount of time that has elapsed
 @param &burnAmount    - The amount of fuel that is going to be burnt


 ***/


void touchdown(double &elapsedTime, double &velocity, double&burnAmount, double &height)
//Defining the function touchdown, which will do some corrections at the point of touchdown in order to
//get accurate final values of height and velocity

{

    double frac;
    //Declares a double type variable named frac
    if (burnAmount == 5)
    // Checks whether the value of burnAmount is equal to 5
    //If it's so then continues, or moves to other conditional statement.

    {
        frac = height / velocity;
    // Computes the value of the variable frac
    }
    else
         // If the previous 'if' and 'else if' statements fail, then the program
       // run through this statement.
    {
        frac = (sqrt((pow(velocity, 2)) + (height*(10 - 2*burnAmount))) - velocity )/( 5- burnAmount);
        // As acceleration due to gravity is set to 5
    // Computes the value of the variable frac
    }
    elapsedTime  = elapsedTime + frac;
    // Computes the value of the variable elapsedTime
    velocity    += (5- burnAmount)*frac;
    // Computes the value of the variable velocity
    height       = 0;
   // Computes the value of the variable height
}


/***
                This function reports the damage based on the final velocity

 @param &os       - The ofstream object
 @param velocity - The velocity of the falling spaceship


 ***/


void finalAnalysis(ostream &os, double velocity)
//Defining the function finalAnalysis, which will report the damage based on the final velocity

{
        if (velocity == 0)
            // Checks whether the value of velocity is equal to 0
        //If it's so then continues, or moves to other conditional statement.
        {
            os << "Congratulations! A perfect landing!!" << endl << "Your license will be renewed...later." <<
endl;
            //Setting output to the output file as directed by the user
            cout << "Congratulations! A perfect landing!!" << endl << "Your license will be renewed...later." <<
endl;
        // Shows the report according to the condition
        }
        else if(velocity > 0 && velocity < 2)
            // Checks whether the value of velocity is greater than 0 and less than 2
        //If it's so then continues, or moves to other conditional statement.
        {
            os << "A little bumpy." << endl;
            cout << "A little bumpy." << endl;
        // Shows the report according to the condition
        }
        else if(velocity >= 2 && velocity < 5)
            // Checks whether the value of velocity is greater than or equal to 2 and less than 5
        //If it's so then continues, or moves to other conditional statement.
        {
            os << "You blew it!!!!!!" << endl << "Your family will be notified...by post." << endl;
            cout << "You blew it!!!!!!" << endl << "Your family will be notified...by post." << endl;
            // Shows the report according to the condition

        }
        else if(velocity >= 5 && velocity < 10)
            // Checks whether the value of velocity is greater than or equal to 5 and less than 10
        //If it's so then continues, or moves to other conditional statement.
        {
            os << "Your ship is a heap of junk !!!!!" << endl << "Your family will be notified...by post." <<
endl;
            cout << "Your ship is a heap of junk !!!!!" << endl << "Your family will be notified...by post." <<
endl;
            // Shows the report according to the condition

        }
        else if(velocity >= 10 && velocity < 30)
            // Checks whether the value of velocity is greater than or equal to 10 and less than 30
        //If it's so then continues, or moves to other conditional statement.
        {
            os << "You blasted a huge crater !!!!!" << endl << "Your family will be notified...by post." << endl
;
            cout << "You blasted a huge crater !!!!!" << endl << "Your family will be notified...by post." <<
endl;
            // Shows the report according to the condition

        }
        else if(velocity >= 30 && velocity < 50)
            // Checks whether the value of velocity is greater than or equal to 30 and less than 50
        //If it's so then continues, or moves to other conditional statement.
        {
            os << "Your ship is a wreck !!!!!" << endl << "Your family will be notified...by post." << endl;
            cout << "Your ship is a wreck !!!!!" << endl << "Your family will be notified...by post." << endl;
            // Shows the report according to the condition

        }
        else
            // If the previous 'if' and 'else if' statements fail, then the program
       // run through this statement.
        {
            os << "You totaled an entire mountain !!!!!" << endl << "Your family will be notified...by post." <<
endl;
            cout << "You totaled an entire mountain !!!!!" << endl << "Your family will be notified...by post."
<< endl;
            // Shows the report according to the condition

        }

}
```