Tutorial letter 201/1/2018
Introduction to Programming II

COS1512

School of Computing

This tutorial letter contains the solutions to Assignment 1

Learn without limits.

UNISA university of south africa

INTRODUCTION

By the time you receive this tutorial letter you should have already completed assignment 1 and we hope that you are well on your way with your studies. This tutorial letter contains the solutions to Assignment 1. You are welcome to e-mail me with any queries at **schoema@unisa.ac.za.** Also take note of the following telephone number and the days on which the lecturer are available in case you have to call.

Mondays	Dr MA Schoeman	011 670 9178
Tuesdays	Dr MA Schoeman	011 670 9178

Allocation of marks

When we mark assignments, we comment on your answers. Many students make the same mistakes and consequently we discuss general problems in the tutorial letters. It is, therefore, important to work through the tutorial letters and to make sure you understand our solutions and where you went wrong. **The maximum number of marks you could obtain for Assignment 1 is 40**. This is converted to a percentage. If you for instance obtained 49 marks for Assignment 1, you received 28/40 * 100 = 70% for Assignment 1. This percentage in turn contributes a weight of 20% to the year mark, as can be seen in the summary of the weights allocated to the assignments for COS1512 below.

Assignment number	Weight
1	20
2	80
3	0

We give the mark allocation for the questions below. For questions 1 – 4 you will not get any marks if you did not include the program code. Or if you only included part of the code, you will get a maximum of 2 marks if the included code is correct. Please note that this is NOT the way exam answers will be marked. If you did not include the output for your program, you will not get full marks for the question. We discuss a possible solution for each question below. Please read through the solution and discussions thoroughly.

The marks you received for question 1 was determined on the following basis:

Question not done	0/10
Question attempted, but the program does not work at all	4/10
A good attempt, but there are a few problems with your answer	8/10
The program works correctly and produces the correct output	10/10

The marks you received for question 2 was determined on the following basis:

Question not done	0/5
Question attempted, but the program does not work at all	2/5
A good attempt, but there are a few problems with your answer	4/5
The program works correctly and produces the correct output	5/5

The marks you received for question 3 was determined on the following basis:
This question was not marked. If you attempted the question, you will get 5 marks. If not, you will get 0 marks. Please go through the solution that we give for question 3 to make sure that you understand how to work with data files.

The marks you received for question 4 was determined on the following basis:

Question not done	0/15
Question attempted, but the program does not work at all	5/15

A good attempt, but there are a few problems with your answer	11/15
The program works correctly and produces the correct output	15/15

The marks you received for question 5 was determined on the following basis:
This question was not marked. If you attempted all the questions, you will get 5 marks. If you have not attempted both questions 5(m) and (n), you will get 2 marks. If you have not attempted one of questions 5(m) or 5(n), you will get 3 marks. If you have not attempted the question at all, you will get 0 marks. Please go through the solution that we give for question 5 to make sure that you understand how to work with pointers.

Solution to Assignment

Question 1 10 marks

For this question you had to write a program to use two overloaded functions, each named `calcFees()`, to determine the tuition fees for a student. Students who repeat a module pay a different fee for the modules which are repeated. The number of modules repeated or taken for the first time, should be taken into account when the fee is calculated.

Overloaded functions need to differ in either the number of parameters and/or the type of parameters supplied to the functions. In this question you had to write an overloaded function (`calcFees()`) with either two or four parameters of type `double`. The question did not clearly state whether the respective fees should be input by the user. We defined two `double` constants `feeFirst` and `feeRepeat` for this purpose, but you could have asked the user to input the fees as well.

```
#include <iostream>
using namespace std;

// overloaded function for tuition fees - first time modules only
// Pre: first time enrolled modules + fees
// Post: returned value is tuition fees
double calcFees (int nrFirstTime, double tuitionFirst);

// overloaded function for tuition fees - first time and repeated modules
// Pre: first time + repeat modules + respective fees
//Post: returned value is tuition fees
double calcFees (int nrFirstTime, const double feeFirst,
                 int nrRepeat, const double feeRepeat);

const double feeFirst = 1050.00;
const double feeRepeat = 900.00;

int main()
{
    char answer;
    int nrFirstTime = 0, nrRepeat = 0;
    double tuitionAmount = 0.00;

    cout << "Does the student repeat any modules? y or n: " << endl;
    cin >> answer;
    if ((answer == 'y') || (answer == 'Y'))
    {
        cout << "How many modules will be repeated? : ";
    cin >> nrRepeat;
    }
    cout << "How many modules are first time modules? : ";
    cin >> nrFirstTime;

    if (nrRepeat == 0)
        tuitionAmount = calcFees (nrFirstTime, feeFirst);
    else
        tuitionAmount = calcFees(nrFirstTime,feeFirst,nrRepeat, feeRepeat);

    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(2);
    cout << endl << "The total tuition fee for this student is : R"
        << tuitionAmount << endl;
    return 0;
}

//overloaded function for first time modules only
double calcFees (int nrFirstTime, const double feeFirst)
{
    return (feeFirst * nrFirstTime);
}

//overloaded function for first time modules and modules that are repeated
double calcFees (int nrFirstTime, const double feeFirst, int nrRepeat,
                 const double feeRepeat)
{
    return (feeFirst * nrFirstTime + feeRepeat * nrRepeat);
}
```

The overloaded functions have the same function name `CalcFees()`.The difference is the number of parameters in these functions. The compiler will use the number of parameters to decide which function to call.

Input and corresponding output version 1:

```
Does the student repeat any modules? y or n:
y
How many modules will be repeated? : 2
How many modules are first time modules? : 4

The total tuition fee for this student is : R6000.00

Process returned 0 (0x0)   execution time : 9.240 s
Press any key to continue.
```

Input and corresponding output version 2:

```
Does the student repeat any modules? y or n:
n
How many modules are first time modules? : 6

The total tuition fee for this student is : R6300.00

Process returned 0 (0x0)   execution time : 5.650 s
Press any key to continue.
```

Question 2 5 marks

In this question you had to write a program to convert time from 24-hour notation to 12-hour notation, for example, 14:25 should be converted to 2:25 PM. The program had to ensure that both the hours and minutes provided as input are valid. We used the `assert` macro as follows:

```
        assert(hour >= 0 && hour <=24);
        assert(minute >= 0 && minute <=59);
```

To use the `assert()` function in your program, you must include the `cassert` header file in your program:

```
#include <cassert>
```

We expected you to check the validity of the input time with the `assert` macro. Although you were expected to use the `assert` statement, the restriction could also be enforced as follows:

```
if((hour < 0 || hour > 24 || minute < 0 || minute > 60)
{
      cout << "Quitting. Invalid 24 hour format time\n";
      exit(0);
}
```

Typically the `assert` macro is used to identify program errors during development. Remember that the `assert()` function evaluates a boolean expression. If the result is 1 (true), the program continues. If the result is 0, the program aborts with an exception. The argument given to `assert` should therefore be chosen so that it holds true only if the program is operating as intended. The macro evaluates the `assert` argument and, if the argument expression is false the program execution is halted. No action is taken if the argument is true, i.e. the program simply continues executing.

You did not have to handle the exception that occurs if the result of the `assert()` function is 0, in your program. We just wanted to see that you did get the exception in your output, when the conditions were not met.

We highlight the use of the `assert()` function in the program listing below, and show the three sets of output when different times are input.

Program listing:

```
//Ass1 question 2
#include <iostream>
#include <string>
#include <cmath>
#include <cstdlib>
#include <cassert>
using namespace std;

//Pre: 24 hour format time entered as hours and minutes ( 2 integers)
//     Required : 24 hour format
//Post: returned 12 hour format

void convertTime(int hours, int minutes);

int main()
{
    int hour, minute;
    char ans;
    do
    {
        cout << "Please enter the hours and minutes separately (e.g. 23 "
             << "56): ";
        cin >> hour >> minute;
        assert(hour >= 0 && hour <=24);
        assert(minute >= 0 && minute <=59);

        cout << "The 12 hour format of " << hour << ":" ;
        if (minute < 10)
            cout << 0;
        cout << minute << " is " ;
        convertTime(hour, minute);
```

The assert function is used to check if the input meets the criteria .If it meets the criteria, the program will continue. If the criteria are not met, the assert function will throw an exception error.

```
        cout << endl<< "Please enter Y/y to convert another time(N/n to "
             << "exit): ";
        cin >> ans;
        cout << endl;
    } while(ans == 'y' || ans == 'Y');

return 0;
}

void convertTime(int hours, int minutes)
{
    if (hours > 12)
        cout << hours - 12 << ":" << minutes << "pm" << endl;
    else cout << hours << ":" << minutes << "am" << endl;
}
```

Input and corresponding output 1:

```
Please enter the hours and minutes separately (e.g. 23 56): 23 56
The 12 hour format of 23:56 is 11:56pm

Please enter Y/y to convert another time(N/n to exit): n



Process returned 0 (0x0)   execution time : 7.690 s
Press any key to continue.
```

Input and corresponding output 2:

```
Please enter the hours and minutes separately (e.g. 23 56): 25 45
Assertion failed: hour >= 0 && hour <=24, file
C:\Unisa\COS1512\2018\Time.cpp, l
ine 23

This application has requested the Runtime to terminate it in an unusual
way.
Please contact the application's support team for more information.

Process returned 3 (0x3)   execution time : 11.530 s
Press any key to continue.
```

Input and corresponding output 3:

```
Please enter the hours and minutes separately (e.g. 23 56): 11 61
Assertion failed: minute >= 0 && minute <=59, file
C:\Unisa\COS1512\2018\Time.cp
p, line 24

This application has requested the Runtime to terminate it in an unusual
way.
Please contact the application's support team for more information.

Process returned 255 (0xFF)   execution time : 11.070 s
Press any key to continue.
```

Question 3 This question has not been marked – 5 marks if you attempted it

Discussion of solution

For this question you had to write a program to prepare a statement for a customer's cheque account at the end of each month, based on data provided in an input file.

The first step is to create the input file. We created the input file `bank.dat` by using the `Code::Blocks` editor and creating a new source file, enter the data, and save it as a file with an extension of `.dat`. You could also have used Notepad. Save your input file in the **same directory** where you save your program.

The contents of the input file (and any output file should the program have produced one or more) can be viewed with `Code::Blocks`. This file is opened with `File | Open Project or File` and by selecting the correct file in the correct directory.

The program has to ask the user for the name of the input file and then read the file, manipulate data and then display output on the console. Therefore the header file for the `fstream` library have to be added to the code with the `#include <fstream>` directive.

We read the input file by opening a file called `bank.dat` and associating the file names with the `ifstream` variables, as shown in the statements below:

```
cout << "Please enter the input file name : "<< endl;
cin  >> inName;
inFile.open(inName.c_str());
```

Note that if you create the input file in a directory different from the one where you save your program, you need to specify the path as well, when specifying the filename, e.g.

```
C:\cos112\datafiles\bank.dat
```

When using a file it is essential to test whether the file is available. The following code segment tests whether the first input file is available before attempting to extract from it:

```
if (!inFile)
{
    cout << "Cannot open file " << inName << " Aborting!" << endl;
    exit(1);
}
```

In the above code segment the program is terminated immediately when `exit(1)` is invoked. We need to add

```
#include <cstdlib>
```

to our program i.e. the header file for the library that contains this function.

The final step is to close the output file:

```
inFile.close();
```

Program Listing:

```
#include <iostream> // for screen/keyboard i/o
```

```
#include <fstream>  // for file
#include <string>   //for string manipulation
#include <cstdlib>  // for exit
#include <iomanip> //defines the manipulator functions
using namespace std;

//Declare contestant variables
const double MIN_BALANCE = 1000.00;
const double SERVICE_CHARGE = 25.00;

int main()
{
    int accNumber;
    double initialBalance;
    double accBalance;
    double amtDeposited = 0.0;
    int noOfDeposits = 0;
    double amtWithdrawn = 0.0;
    int noOfWithdrawals = 0;
    double interestPaid;
    char transactionCode;
    double transactionAmt;
    double bank_cost = 0;
    ifstream inFile;
    string inName;

    cout << "Please enter the input file name : "<< endl;
    cin  >> inName;
    inFile.open(inName.c_str());

    if (!inFile)
    {
        cout << "Cannot open file " << inName << " Aborting!" << endl;
        exit(1);
    }

    inFile>>accNumber>>initialBalance;
    accBalance = initialBalance;

    //Output Results heading
    cout << setprecision(2);
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout <<"Account Number: " << accNumber << endl << endl;
    cout << "Opening Balance Balance : R " << initialBalance << endl
         << endl;
    cout << left << setw(15) << "Transaction";
    cout << right << setw(15) << "Amount";
    cout << right << setw(15) << "Balance";
    cout << right << setw(15) << "Bank costs" << endl;

    while(inFile >> transactionCode >> transactionAmt)
    {
        switch(transactionCode)
        {
            case 'D':
            case 'd': {accBalance = accBalance + transactionAmt;
                        noOfDeposits++;
```

One gets input from a file into your program, or send output to a file from your program by using streams, or special objects as they are called in C++. The type for input-variable streams is named i f st r eam, and for outputvariable streams, of st r eam. One connects the object to the file by opening the file, as is done in the code. We include the f st r eam header file as well.
Please see section 6.1 in Savitch for more info.

```
                        cout << left << setw(15) << "Deposit";
                        cout << right << setw(15) << transactionAmt
                             << "Ct";
                        cout << right << setw(13) << accBalance;

                        if(accBalance < MIN_BALANCE)
                        {
                            bank_cost = bank_cost + SERVICE_CHARGE;
                            cout << right << setw(15) << SERVICE_CHARGE
                                 << endl;
                        }
                        else cout << endl;
                        }
                        break;
            case 'I':
            case 'i': accBalance = accBalance + transactionAmt;
                        cout << left << setw(15) << "Interest";
                        cout << right << setw(15) << transactionAmt;
                        cout << right << setw(15) << accBalance
                             << endl;
                        break;
            case 'W':
            case 'w': {accBalance = accBalance - transactionAmt;
                        noOfWithdrawals ++;
                        cout << left << setw(15) << "Withdrawal";
                        cout << right << setw(15) << transactionAmt;
                        cout << right << setw(15) << accBalance;
                        if(accBalance < MIN_BALANCE)
                        {
                            bank_cost = bank_cost + SERVICE_CHARGE;
                            cout << right << setw(15) << SERVICE_CHARGE
                                 << endl;
                        }
                        else cout << endl;
                        }
                        break;
            default:    cout << "Invalid transaction code" << endl;
        }
    }

    accBalance = accBalance - bank_cost;
    cout << "\nTotal of Banking Costs R" << bank_cost;
    cout << "\nClosing Balance R " << accBalance << endl;

    inFile.close();
    return 0;
}
```

Inputfile (bank.dat):

```
46780976 1750.40
W 1250.00
D 200.00
W 75.00
W 375.00
D 1200.00
I 5.50
W 400.00
```

```
W 600.00
D 450.50
W 35.65
```

Output:

```
Please enter the input file name :
bank.dat
Account Number: 46780976

Opening Balance Balance : R 1750.40

Transaction             Amount        Balance     Bank costs
Withdrawal             1250.00         500.40          25.00
Deposit                 200.00Ct       700.40          25.00
Withdrawal               75.00         625.40          25.00
Withdrawal              375.00         250.40          25.00
Deposit                1200.00Ct      1450.40
Interest                  5.50        1455.90
Withdrawal              400.00        1055.90
Withdrawal              600.00         455.90          25.00
Deposit                 450.50Ct       906.40          25.00
Withdrawal               35.65         870.75          25.00

Total of Banking Costs R175.00
Closing Balance R 695.75

Process returned 0 (0x0)   execution time : 4.210 s
Press any key to continue.
```

Question 4 15 marks

Discussion:

The purpose of the program is to read a file character by character. The input file contains a paragraph in which the numbers 0 to 7 must be changed to characters.

We did not write a function to perform this task – it is performed by the `main` function. We did, however write a function to read and display the input file, as well as the output file that was created from the input file. It is good programming practice to put the code that does all the reading and writing together.

When implementing our solution, once again the first step is to create the input file. Since the purpose of the program is to process files, the `#include <fstream>` and `#include <cstdlib>` directives have to be included in the program in order to use the files and to test that they exist before they are used. The names of the input and output files are requested from the user. We did not specify that this had to be done, so you could also have initialized the file names in your program.

In this program we process the input file as a text file. A text file is typically processed in the following way:

```
      char ch;
      infile.get(ch);
      while (!infile.eof())
      {
            //process ch
            infile.get(ch);
      }
```

Compare this to the typical way to read a file containing values that should be processed as `int`, `string` or `float` (as in Question 3), e.g. a file containing one `int` followed by a `string` on each line:

```
      int value;
      string name;
      while (infile >> value >> name)
      {
      //process value and name
      }
```

After having created the output file, we added some code to read the input file and display the contents, as well as some code to do the same for the output file. Note that after the output file is created and closed, it now acts as an input file if we want to read and display its contents. We therefore need an `ifstream` definition. For this purpose we defined:

```
      ifstream indisplay;
      ifstream outdisplay;
```

The first declaration will represent the original input file. The second declaration will represent the created output file as an input file.

Program listing:

Note that we use an array `char convert[8] = {‘s’,’g’,’o’,’y’,’v’,’n’,’f’,’j’};` to convert the digits 0 to 7 to the corresponding characters with the following code:

```
      if (ch >= '0' && ch < '8')
      {
      index = int(ch – ‘0’);
      outfile << convert[index];
      }
```

We could have used a switch statement to do the same, e.g.:

```
         if (ch >= '0' && ch < '8')
         {
             switch (ch)
             {
                 case '0':
                     ch = 's'; break;
                 case '1':
                     ch = 'g'; break;
                 case '2':
                     ch = 'o'; break;
                 case '3':
                     ch = 'y'; break;
                 case '4':
                     ch = 'v'; break;
                 case '5':
                     ch = 'n'; break;
                 case '6':
                     ch = 'f'; break;
                 case '7':
                     ch = 'j'; break;
                 default: ch = ch;
             }
```

```
            outfile << ch;
        }
```

Program code:

```
//Ass 1 question 4
#include <iostream> // for screen/keyboard i/o
#include <fstream> // for file
#include <cstdlib> // for exit using namespace std;
#include <cstring>
using namespace std;
// Precondition:
// The input file is a text file.
// The input file has been opened.
//
// Postcondition:
// The output file is a text file.
// The output file has been opened.
// Output file will be similar to input file except for all numbers 0
// to 7 that will be replaced with the character as specified in the
// question

void checkFile(ifstream& infile)
{
    char ch;
    infile.get(ch);
    while(!infile.eof())
    {
        cout << ch;
        infile.get(ch);
    }
}

int main()
{
    ifstream infile;
    ofstream outfile;
    ifstream indisplay;
    ifstream outdisplay;
    string inName, outName;

    cout << endl << "Enter the input file name. " << endl;
    cin >> inName;
    infile.open(inName.c_str());
    if (infile.fail())
    {
        cout << "Cannot open file " << inName << " Aborting!" << endl;
        exit(1);
    }

    cout << endl << "Enter the output file name. " << endl
        << "WARNING: ANY EXISTING FILE WITH THIS NAME WILL"
        <<" BE ERASED." << endl;
    cin >> outName;
    outfile.open(outName.c_str());
    if (outfile.fail())
    {
        cout << "Cannot open file " << outName << " Aborting!" << endl;
        exit(1);
    }

    cout << endl;
    char ch;

    char convert[8] = {'s','g','o','y','v','n','f','j'};
    int index;
    infile.get(ch);
    while(!infile.eof())
    {
        if (ch >= '0' && ch < '8')
        {
            index = int(ch - '0');
            outfile << convert[index];
        }
        else
            outfile << ch;
        infile.get(ch);
    } //end while !infile.eof

    infile.close();
    outfile.close();

    //This part was not required, but it is always a good idea to read the
    // file that was created to make sure it is correct.
    //We first read the original input file, and display its content
    indisplay.open(inName.c_str());
    if (indisplay.fail())
    {
        cout << "Cannot open file " << inName << " Aborting!" << endl;
        exit(1);
    }

    cout << endl << "The contents of the input file is : "<< endl << endl;
    checkFile(indisplay);
    indisplay.close();

    //Now we read the file that was created as an input file and display
    // the content
    outdisplay.open(outName.c_str());
    if (outdisplay.fail())
    {
        cout << "Cannot open file " << outName << " Aborting!" << endl;
        exit(1);
    }
    cout << endl << endl <<"The contents of the output file is : "
        << endl << endl;
    checkFile(outdisplay);
    outdisplay.close();
    cout << endl;

    return 0;
}
```

Output:

```
Enter the input file name.
activity.txt
```

```
Enter the output file name.
WARNING: ANY EXISTING FILE WITH THIS NAME WILL BE ERASED.
competition.txt


The contents of the input file is :

We h2pe that 32u e5723ed the acti4it3. A6ter 32u ha4e c2mpleted the
acti4it3,0e5d 32ur re0ult t2: The Acti4it3 C2mpetiti25, Bett3 Da4i0 0treet
99, Auckla5d Park, 8989, a5d 0ta5d a cha5ce t2 wi5 a hamper c250i0ti51 26
c2l2uri51 a5d acti4it3b22k0, c2l2uri51 pe5cil0 a5d pe50.


The contents of the output file is :

We hope that you enjoyed the activity. After you have completed the
activity, send your result to: The Activity Competition, Betty Davis street
99, Auckland Park, 8989, and stand a chance to win a hamper consisting of
colouring and activity books, colouring pencils and pens.


Process returned 0 (0x0)   execution time : 11.500 s
Press any key to continue.
```

Question 5 This question has not been marked – 5 marks if you attempted it

(a) A pointer is the memory address of a variable. A variable's address can be thought of as 'pointing' to the variable. See section 9.1 in Savitch.

(b) The deferencing operator is the *operator (the asterisk) used in front of a pointer variable. It dereferences the pointer variable to produce the variable to which the pointer is pointing to.

(c) Assuming both `p1` and `p2` have been declared as pointers, i.e. as follows:
`int *p1, *p2;`
In the assignment statement `p1 = p2`, the value of one pointer (`p2`) is assigned to another pointer (`p1`) so that the two pointers point to the same memory location. Basically you are using the actual pointers (addresses of memory locations). With the assignment statement `*p1 = *p2`, you are using the actual variables to which the pointers are pointing to so that both variables will have the same value.

(d) A dynamic variable is a variable that is created and destroyed during the execution of the program. It is created using the `new` operator.

(e) The `new` operator produces a new, nameless variable, with a specified data type and returns a pointer that point to this new variable. This means that the only way the program can access the variable is through the pointer pointing to it.

(f) The `delete` operator eliminates (releases or erases) a dynamic variable and returns the memory that the dynamic variable occupied to the freestore. It releases the memory so that it can be used for the creation of new dynamic variables.

(g) The freestore (also called the heap) is a special area in memory that is reserved to be used for dynamic variables.

(h) Dynamic variables are created in a reserved space in memory (the freestore or heap). They are created and destroyed while the program is running. Automatic variables are automatically created when the function in which they are declared is called and automatically destroyed when the function ends. The ordinary variables we use in the programs we write for COS1512 are automatic variables.

(i) A dynamic array is an array whose size is not specified when it is declared in the program. Instead, its size is determined while the program is running.

(j) They are flexible in terms of size since the size of the array can be specified during the run time of the program. This avoids the problem of specifying an array that is too small (not having enough elements) or too big (wasting computer memory space).

(k) An array variable is a pointer variable that points to the first indexed variable in an array.

(l) i. `typedef int* int_ptr;`

ii. `int_ptr p1;`
OR
```
int *p1;
```

iii. `p1 =new int;`

iv. `*p1 = 23;`

v. `int a;`

vi. `p1 = &a;`

vii. `delete p1;`
The value of the variable that `p1` is pointing to is now undefined, since that memory location has been released.

(m)
i. `typedef int* int_ptr;`

ii. `int_ptr p2;`

iii.
```
int nrElements;
cout << "How many elements should be in the array? " << endl;
cin >> nrElements;
```

iv. `p2 = new int[nrElements];`

v. `int a[500];`

vi.
```
for (int i = 0; i <= nrElements; i++)
      a[i] = p2[i];
```

(n)
Program:
```
#include <iostream>
using namespace std;
int main()
{
    typedef int* IntArrayPtr;
    IntArrayPtr examMarks;
    int size, total, average;
    cout << " How many exam marks will be entered? ";
    cin >> size;

    examMarks = new int[size];
    cout << "Please enter " << size << " exam marks: " <<endl;
    for (int i = 0; i < size; i ++)
        cin >> examMarks[i];

    total = 0;
    for (int i = 0; i < size; i ++)
        total = total + examMarks[i] ;

    average = int(total/size);

    cout << endl << "The average exam mark is " << average << endl;

    delete [] examMarks;
    return 0;
}
```

Output:
```
How many exam marks will be entered? 3
Please enter 3 exam marks:
50
60
70

The average exam mark is 60

Process returned 0 (0x0)   execution time : 6.070 s
Press any key to continue.
```