Tutorial letter 201/2/2017
Introduction to Programming II

COS1512

School of Computing

This tutorial letter contains the solutions to Assignment 1

IMPORTANT INFORMATION:
Please activate your *myUnisa* and *my*Life email addresses and ensure you have regular access to the *my*Unisa module site COS1512-2015-S2 as well as your e-tutor group site.

Due to regulatory requirements imposed by the Department of National Education the following apply:

To be considered for examination admission in COS1512, a student must meet the following requirement:

Submit assignment 1 or assignment 2 BEFORE 8 September 2017.

Note: This is a blended online module, and therefore your module is available on myUnisa. However, in order to support you in your learning process, you will also receive some study materials in printed format. Please visit the COS1512 course website on myUnisa at least twice a week.

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 80**. This is converted to a percentage. If you for instance obtained 60 marks for Assignment 1, you received 60/80 * 100 = 75% 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. 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 3/10
A good attempt, but there are a few problems with your answer 7/10
The program works correctly and produces the correct output 10/10

The marks you received for question 2 was determined on the following basis:
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 2 to make sure that you understand the assert() function.

The marks you received for questions 3 and 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 10/15
The program works correctly and produces the correct output 15/15

The marks you received for question 5 was determined on the following basis:
One mark each for question 5(a) – (k) that is max 11 marks
One mark each for question 5(l)(i)-(xii) that is max 13 marks

One mark each for question 5(m)(i)-(vi) that is max 6 marks
5 marks for question 5(n)
Total number of marks for question 5 is max **35** marks.

Question 1 **10 marks**

In this question, you were required to write a C++ program to determine whether a ticket number is a winning number, and calculate the cash prize for the ticket number. For a ticket number divisible by 100, for males with a ticket number greater than 30000, the prize money is the ticket number divided by 90.00. For a ticket number divisible by 100, for females with a ticket number greater than 20000, the prize money is the ticket number divided by 80.00. Alternatively, if the ticket is not divisible by 100, but divisible by 7 and divisible by 6, the user must be asked to input his/her age. The prize money is then calculated as follows: for up to an age of 21, the prize money is the age multiplied by 40. For an age over 21 the prize money is the age multiplied by 30. If the ticket number does not adhere to the rules, a message should be displayed to say that the ticket is not a winning ticket.

You had to overload function `getPrize()`.

Overloading is a feature in C++ that allows the programmer to define more than one function with the same name, but with either a different number of function parameters, or with parameters of different types. The compiler will use either the number of parameters, or the type of parameters to choose which function to call. The first overloaded function will calculate the prize money for ticket numbers divisible by 100, and the second overloaded function will calculate the prize money for ticket numbers not divisible by 100. We show three different outputs for the program listing below to show what happens when each of the two overloaded functions are called, as well as in example of an invalid ticket number.

Program listing:

```
1  #include <iostream>
2  using namespace std;
3  double getPrize(int ticket, char gender)
4  {
5     double prize = 0.00;
6     if (gender == 'm' && ticket > 30000)
7     {
8         prize = (double)ticket / 90.00;
9     }
10    if (gender == 'f' && ticket > 20000)
11    {

12        prize = (double)ticket / 80.00;
13    }
14    return (prize);
15 }
16 double getPrize(int ticket, int age)
17 {
18    if (age <= 21)
19       return (age *40);
20    else if (age > 21)
21       return (age * 30);
22 }

23 int main()
```

The functions on lines 3 and 16 have the same function name `GetPrize()`. The function is overloaded. The difference in this case is the types of parameters in these functions. Both functions have two parameters, in line 3 the function has an `int` and a `char` parameter while in line 16, the function has two `int` parameters. The compiler will use the types of the parameters to decide which function to call. For instance, when the function is called with two `int` parameters, it will use the function that starts at line 16. When `GetPrize()` is called with an `int` parameter and a `char` parameter it will use the function starting at line 3.

```
24 {

25    int ticketnumber, age;
26    char gender;
27    double prize  = 0.00;
28    cout << "Please enter your ticket number: ";
29    cin >> ticketnumber;
30    cout.setf(ios::fixed);
31    cout.precision(2);

32    if ((ticketnumber % 100) == 0)
33    {
34       cout << "Please enter your gender (m or f): ";
35       cin >> gender;
36       prize = getPrize(ticketnumber, gender);
37       cout << "Your prize is R" << prize << endl;
38    }
39    else if (((ticketnumber % 7)==0) && ((ticketnumber % 6) == 0))
40    {
41       cout << "Please enter your age: ";
42       cin >> age;
43       prize = getPrize(ticketnumber, age);
44       cout << "Your prize is R" << prize << endl;
45    }
46    else
47        cout << "Sorry, your ticket number has not won any prize" <<
      endl;
48    return 0;
49 }
```

Output 1:

```
Please enter your ticket number: 25346
Sorry, your ticket number has not won any prize

Process returned 0 (0x0)   execution time : 13.250 s
Press any key to continue.
```

Output 2:

```
Please enter your ticket number: 25000
Please enter your gender (m or f): f
Your prize is R312.50

Process returned 0 (0x0)   execution time : 15.860 s
Press any key to continue.
```

Output 3:

```
Please enter your ticket number: 42042
Please enter your age: 34
Your prize is R1020.00

Process returned 0 (0x0)   execution time : 29.490 s
Press any key to continue.
```

Question 2 This question has not been marked – 5 marks if you attempted it

In this question you had to convert two times in 24h00 format (hh mm ss) to seconds, get the difference between the two times and display the result in 24h00 time format again. You also had to make sure that the times are valid. We used the `assert` macro as follows:

```
cout << "Please enter the first time as 3 fields in the format hh mm ss"
     << "(e.g. 08 20 45) :" ;
    cin >> hours >> minutes >> seconds;
    assert(hours >= 0 && hours <= 24);
    assert(minutes >=0 && minutes <= 59);
    assert(seconds >= 0 && seconds <= 59);
    if (hours == 24)
        assert(minutes == 0 && seconds == 0);
```

Note the definition of `assert()` that had to be included:
`#include <cassert>`

We expected you to check the validity of the times with the `assert` macro. This is of course also possible without the `assert` macro.

Program listing:

```
1    //Ass 1 question 2
2    #include <iostream>
3    #include <cassert>
4    using namespace std;
5    int calcSeconds(int hrs, int mins, int secs)
6    {
7       return (hrs * 3600 + mins * 60 + secs);
8    }
9    void calcTime(int &hrs, int &mins, int &secs)
10   {
11      hrs = secs / 3600;
12      mins = (secs % 3600) / 60;
13      secs = (secs % 3600) % 60;
14   }
15   int main()
16   {
17      int hours, minutes, seconds, difference, sec1, sec2;
18      cout << "Please enter the first time as 3 fields in the format "
             << "hh mm ss(e.g. 08 20 45): " ;
19      cin >> hours >> minutes >> seconds;
20      assert(hours >= 0 && hours <= 24);
21      assert(minutes >=0 && minutes <= 59);
22      assert(seconds >= 0 && seconds <= 59);
23      if (hours == 24)
           assert(minutes == 0 && seconds == 0);
24      sec1 = calcSeconds(hours, minutes, seconds);
25      cout << "Please enter the second time as "
             << "3 fields in the format "
            << "hh mm ss (e.g. 08 20 45): " ;
26      cin >> hours >> minutes >> seconds;
27      assert(hours >= 0 && hours <= 24);
```

The `assert` function is used to check if the input meets the criteria. The inputs in this case are the hours, minutes and seconds. If it meets the criteria, the program will continue. If the criteria are not met, the `assert` function will throw an exception error.

```
28      assert(minutes >=0 && minutes <= 59);
29      assert(seconds >= 0 && seconds <= 59);
30      if (hours == 24)
           assert(minutes == 0 && seconds == 0);
31      sec2 = calcSeconds(hours, minutes, seconds);
32      if (sec1 > sec2)
           difference = sec1 - sec2;
33      else
           difference = sec2 - sec1;
34      hours = minutes = 0;
35      calcTime(hours, minutes, difference);
36      cout << endl <<"The difference between the two times is "
             << hours <<":"<<minutes<<":"<< difference <<endl;
37      return 0;
38   }
```

Output for first set of test data

```
Please enter the first time as 3 fields (e.g. 08 20 45): 20 43 22
Please enter the second time as 3 fields (e.g. 08 20 45): 02 04 55

The difference between the two times is 18:38:27
```

Output for second set of test data

```
Please enter the first time as 3 fields in the format hh mm ss(e.g. 08 20 45):
23 59 61
Assertion failed: seconds >= 0 && seconds <= 59, file C:\Unisa\COS1512\2017\time
s.cpp, line 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 : 19.610 s
Press any key to continue.
```

Question 3	**15 marks**

Discussion

This program had to read numbers from a file, and determine the median. The median is the middle element of the file if there are an odd number of entries, or the average of the middle two elements if the file has an even number of entries. You had to open the file, count the number of entries, close the file and calculate the position of the middle of the file, open the file again, count up to the entries before the middle element and extract the middle element, or if there are an even number of entries in the file, count up to the middle two elements and calculate the median as the average of the two middle elements.

The first step is to create the input file. We created the two input files `File1.dat` and `File2.dat` by using the DevC++ editor and creating a new source file, entering the data, and saving it as a file with an extension of `.dat`. (You could also have used Notepad.) Save your input files in the **same directory** where you save your program.

The header file for the `fstream` library has to be added to the code with the `#include <fstream>` directive for file processing.

Though the question did not specify that the contents of the input file should be displayed on the console window, we do so in our solution (see lines 73 to 77).

We open the input files `File1.dat` or `File2.dat`, by obtaining the file name from the user and associating the file name with the `ifstream` variables, as shown in the statements (lines 12 to 13 and 16 to 20 in the program listing) below:

```
12    ifstream fin;
13      string inName;

16    //Open the input file.
17      cout << endl << "Enter the input file name: ";
18      cin >> inName;

19      cout << "Opening the input file..." << endl << endl;
20      fin.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:\unisa\cos112\datafiles\File1.dat
```

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

```
21    if (!fin)
22    {
23    cout << "Error accessing input"
24    << " file." << endl;
25    exit(1);
26    }
```

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 the `exit()` function.

To find the median you had to open the input file, count the number of entries, close the file and calculate the position of the middle of the file, open the file again, count up to the entries you need, and calculate the middle. Before we can open the input file for a second time to extract entries up to the median, it must first be closed and then opened again.

```
34    fin.close();
35    fin.open(inName.c_str());
```

We also check that it is available in lines 37 to 40.

Some students may have used arrays to extract and count all the entries from the file and then obtain the median from the array once its position has been calculated. Using arrays has the advantage of immediate access to the entries without having to close and open the file again. On the other hand, should one not know how many entries there are in the file, the array can easily be too small to accommodate all entries or too large, occupying unnecessary memory space. A file has the advantage of storing an unknown number of entries which may also be a very large number of entries. The disadvantage of a file is that should a specific entry be required, all the entries before that must be extracted one by one to reach the desired entry.

Program Listing:

```
1   //Assignment 1 Question 3
2   //Find the median of a list of numbers in a file.
```

```
3   #include <iostream>
4   #include <fstream>
5   #include <cstdlib>
6   #include <cmath>
7   using namespace std;

8   void countNrs(ifstream& in_stream, int& count);
9   //Calculates number of entries in a file.

10  int main()
11  {
12      ifstream fin;
13      string inName;
14      int next, next1, nrOfEntries, count;
15      double median;
16      bool oddNrOfEntries = true; //marker set to assuming the file has an odd
                                    //number of entries

17      //Open the input file.
18      cout << endl << "Enter the input file name: ";
19      cin >> inName;

20      cout << "Opening the input file..." << endl << endl;
21      fin.open(inName.c_str());
22      if (!fin)
23      {
24          cout << "Error accessing input"
25          << " file." << endl;
26          exit(1);
27      }

28      //Count the number of the entries in the file.
29      countNrs(fin, nrOfEntries);
30      cout << "Nr of entries = " << nrOfEntries << endl;

31      if ((nrOfEntries % 2) == 0) //change marker if file
32                                  //has an even number of entries
33      oddNrOfEntries = false;

34      //Reset to the beginning of the file.
35      fin.close();
36      fin.open(inName.c_str());
37      if (!fin)
38      {
39          cout << "Error accessing input file." << endl;
40          exit(1);
41      }

42      //Read entries and count up to the median of the file.
43      count = 0;
44      while(count < nrOfEntries/2 && !fin.eof())
45      {
46          fin >> next;
47          count++;
48      }

49      if (oddNrOfEntries) //the next entry will be the median
50          fin >> median;
51      else     //the average of the previous entry and the next two entry will be
                 //the median
52      {
53         fin >> next1;
```

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 `ifstream`, and for outputvariable streams, `ofstream`. One connects the object to the file by opening the file, as is done in the code. We include the `fstream` header file as well.
Please see section 6.1 of Savitch for more information.

```
54        median = (double(next + next1))/2; //convert the sum of the two middle
       //entries to double to allow double division by 2 to get the correct median
55     }

56     //Close the file.
57     fin.close();

58     cout << "The median of the numbers in the file is "
59          << median << "." << endl;

60     return 0;
61  }

62  void countNrs(ifstream& in_stream, int &count) // count nr of entries in file
                                                  //and display contents of file
63  {
64     int next;
65     count = 0;
66     cout << "Contents of input file:" << endl;
67     while(in_stream >> next)
68     {
69         count++;
70         cout << next << " ";
71     }
72     cout << endl << endl;
73  }
```

Output for first data set (odd number of entries):

```
Enter the input file name: File1.dat
Opening the input file...

Contents of input file:
1 4 6 12 14 18 29 33 37 40 45 47 49 51 55 56 59 60 63

Nr of entries = 19
The median of the numbers in the file is 40.

Process returned 0 (0x0)   execution time : 4.989 s
Press any key to continue.
```

Output for second data set (even number of entries):

```
Enter the input file name: File2.dat
Opening the input file...

Contents of input file:
3 6 7 9 13 16 19 21 26 33 39 41 47 51 58 65 77 80

Nr of entries = 18
The median of the numbers in the file is 29.50.

Process returned 0 (0x0)   execution time : 4.540 s
Press any key to continue.
```

Question 4	**15 marks**

Discussion:
The purpose of the program is to read a file character by character to encrypt or decrypt it. Encryption is done by replacing each character with its successor (replacing e.g. “a” with “b”) while decryption is done by replacing each character by its predecessor (replacing e.g. “b” with “a”).

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.

In this program we process the input file as a text file (see section 6.3 in Savitch). 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, in line 23 and 25, we defined:

```
ifstream infile; //to work with the input file
ifstream indisplay // to display the original input file
ifstream outdisplay;//to display the output file
```

The first declaration will represent the original input file. The second declaration is used when we display the original input file. The third declaration will represent the created output file as an input file.

Note that when file streams are passed as arguments to a function, they need to be reference parameters for the function, since they will be modified, as can be seen from the header for the function `checkFile` in line 6:

```
void checkFile(ifstream& infile).
```

The file streams should be passed by reference rather than by value since the internal state of a file stream object may change with an open operation even if the contents of the file have not changed.

When you declare a reference parameter, the function call will pass the *memory address* of the actual parameter, instead of copying the parameter value into the formal parameter.

Also note that we have to close both the input file and the output file (line 64 and 65 in the program listing below) before we can open them again to display their contents on the screen (see line 91 to line 111 in the program listing below).

The output file can also be viewed with DevC++. This file is opened with `File | Open Project or File` and by selecting the correct file in the correct directory.

Program listing :

```
1    #include <iostream> // for screen/keyboard i/o
2    #include <fstream> // for file
3    #include <cstdlib> // for exit
4    #include <cassert> //for using assert
5    // Precondition:
6    // The input file is a text file.
7    // The input file has been opened.
8    //
9    // Postcondition:
10   // The output file is a text file.
11   // The output file has been opened.

12   using namespace std;

13   void checkFile(ifstream& infile)
14   {
15      char ch;
16      infile.get(ch);
17      while(!infile.eof())
18      {
          cout << ch;
          infile.get(ch);
19      }
20   }

21   int main()
22   {
23      ifstream infile;
24      ofstream outfile;
25      ifstream indisplay;
26      ifstream outdisplay;
27      string inName, outName;
28      char choice;
29      char ch;
30      cout << "Choose 1 if you want to encrypt a file and 2 if you want "
             << "to decrypt a file ";
31      cin >> choice;
32      assert (choice == '1'|| choice == '2');

33      cout << endl << "Enter the input file name. " << endl;
34      cin >> inName;
35      cout << endl << "Enter the output file name. " << endl
             << "WARNING: ANY EXISTING FILE WITH THIS NAME WILL"
             <<" BE ERASED." << endl;
36      cin >> outName;
```

```
37      infile.open(inName.c_str());
38      if (infile.fail())
39      {
40          cout << "Cannot open file "
                 << inName << " Aborting!" << endl;
            exit(1);
42       }

43       outfile.open(outName.c_str());
44       if (outfile.fail())
45       {
46           cout << "Cannot open file "
                      << outName << " Aborting!" << endl;
47          exit(1);
48       }

49       if (choice == '1')
50       {
51          infile.get(ch);
52          cout << endl;
53          while(!infile.eof())
54          {
55             if ((ch >= 'a' && ch <= 'y')||(ch >= 'A' && ch <= 'Y'))
56                 outfile << char (ch + 1);
57             else if (ch == 'z')
58                 outfile << 'a';
59             else if (ch == 'Z')
60                 outfile << 'A';
61             else outfile << ch;
62             infile.get(ch);
63          }
64          infile.close();
65          outfile.close();
66

67       else
68       {
69          infile.get(ch);
70          cout << endl;
71          while(!infile.eof())
72          {
73             if ((ch >= 'b' && ch <= 'z')||(ch >= 'B' && ch <= 'Z'))
74                 outfile << char (ch - 1);
75             else if (ch == 'a')
76                 outfile << 'z';
77             else if (ch == 'A')
78                 outfile << 'Z';
79             else outfile << ch;
80             infile.get(ch);
81       }
82    infile.close();
83    outfile.close();
90    }

91    //We first read the original input file, and display its content
92     indisplay.open(inName.c_str());
93     if (indisplay.fail())
94     {
```

```
95      cout << "Cannot open file "
             << inName << " Aborting!" << endl;
96      exit(1);
97    }
98    cout << endl << "The contents of the input file is : "<< endl
          << endl;
99   checkFile(indisplay);
100  indisplay.close();

101  //Now we read the file that was created as the output file
102  //and display the content

103  outdisplay.open(outName.c_str());
104  if (outdisplay.fail())
105  {
106     cout << "Cannot open file "
             << outName << " Aborting!" << endl;
107     exit(1);
108   }
109   cout << endl<<endl<< "The contents of the output file is : "<< endl
          << endl;
110   checkFile(outdisplay);
111   outdisplay.close();

112   return 0;
113   }
```

The first time we ran this program, we encrypted a file (option 1):

Output:

```
Choose 1 if you want to encrypt a file and 2 if you want to decrypt a file 1

Enter the input file name.
DearJulia.txt

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


The contents of the input file is :

Dear Julia,
You are the most beautiful girl that I have ever seen. I was wondering if
you would like to come and visit me. My mother will make us pancakes with
ice cream. My dog, Bella, just had three beautiful puppies. Mom says I may
only keep one of them. I would like you to help me choose one, because they
are all so cute and adorable. And just because you are my special friend,
you may also have one if you want.
Your friend,
Hector.


The contents of the output file is :
```

```
Efbs Kvmjb,
Zpv bsf uif nptu cfbvujgvm hjsm uibu J ibwf fwfs tffo. J xbt xpoefsjoh jg
zpv xpvme mjlf up dpnf boe wjtju nf. Nz npuifs xjmm nblf vt qbodblft xjui
jdf dsfbn. Nz eph, Cfmmb, kvtu ibe uisff cfbvujgvm qvqqjft. Npn tbzt J nbz
pomz lffq pof pg uifn. J xpvme mjlf zpv up ifmq nf dipptf pof, cfdbvtf uifz
bsf bmm tp dvuf boe bepsbcmf. Boe kvtu cfdbvtf zpv bsf nz tqfdjbm gsjfoe,
zpv nbz bmtp ibwf pof jg zpv xbou.
Zpvs gsjfoe,
Ifdups.

Process returned 0 (0x0)   execution time : 18.220 s
Press any key to continue.
```

The second time we ran this program we, decrypted the file (output.txt) we have encrypted during the first run (option 2). The decrypted file (the output of this run) therefore looks the same as the file we encrypted in the first run (the input for the firs run).

Output 2

```
Choose 1 if you want to encrypt a file and 2 if you want to decrypt a file
2

Enter the input file name.
output.txt

Enter the output file name.
WARNING: ANY EXISTING FILE WITH THIS NAME WILL BE ERASED.
output2.txt


The contents of the input file is :

Efbs Kvmjb,
Zpv bsf uif nptu cfbvujgvm hjsm uibu J ibwf fwfs tffo. J xbt xpoefsjoh jg
zpv xpvme mjlf up dpnf boe wjtju nf. Nz npuifs xjmm nblf vt qbodblft xjui
jdf dsfbn. Nz eph, Cfmmb, kvtu ibe uisff cfbvujgvm qvqqjft. Npn tbzt J nbz
pomz lffq pof pg uifn. J xpvme mjlf zpv up ifmq nf dipptf pof, cfdbvtf uifz
bsf bmm tp dvuf boe bepsbcmf. Boe kvtu cfdbvtf zpv bsf nz tqfdjbm gsjfoe,
zpv nbz bmtp ibwf pof jg zpv xbou.
Zpvs gsjfoe,
Ifdups.


The contents of the output file is :

Dear Julia,
You are the most beautiful girl that I have ever seen. I was wondering if
you would like to come and visit me. My mother will make us pancakes with
ice cream. My dog, Bella, just had three beautiful puppies. Mom says I may
only keep one of them. I would like you to help me choose one, because they
are all so cute and adorable. And just because you are my special friend,
```

```
you may also have one if you want.
Your friend,
Hector.

Process returned 0 (0x0)   execution time : 21.230 s
Press any key to continue.
```

Question 5 **35 marks**

(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. (1)

(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. (1)

(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`). 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. (1)

(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. (1)

(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. (1)

(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. (1)

(g) The freestore (also called the heap) is a special area in memory that is reserved to be used for dynamic variables. (1)

(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. (2)

(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. (1)

(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). (1)

(k) An array variable is a pointer variable that points to the first indexed variable in an array. (1)

(l) i. `double *fPtr1, *fPtr2;` (1)

ii. `fPtr2 = &salary;` (1)

iii. `fPtr1 = &increase;` (1)

iv.
```
cout << "The address of the object pointed to by fPtr1 is "
      <<fPtr1 ;
```
(1)

v.
```
cout << "The value of the object pointed to by fPtr2 "<< *fPtr2;
```
(1)

vi. `fPtr2 = new double;` (1)

vii.
```
if (salary > 4200.00)
     *fPtr2 += *fPtr1;
```
(1)

viii. `fPtr1 = fPtr2;` (1)

ix. `delete fPtr2;` The value of the variable that is `fPtr1` is pointing to, is undefined since both `fPtr1` and `fPtr2` were pointing to the same memory location. (1)

x. `typedef double* DoublePtr;` (1)

xi. `DoublePtr pd;`
(1)

xii `pd = new double[20];` (1)

xii `delete [] pd;` (1)

(m) For this question, by the answer to each sub-question is followed by a comment, indicating the sub-question in the program listing below. We also show the output for the program.

Program listing:
```
#include <iostream>
using namespace std;
int main()
{
    int count;                          //(i)                    (1)
    cout << "please enter a value: ";
    cin >> count;

    int *p1;                            //(ii)                   (1)

    p1 = new int [count];               //(iii)                  (1)
```

```
        for (int i = 0; i < count; i ++)   //(iv)                    (1)
           p1[i] = i;

        cout << "The elements in the array that p1 is pointing to have"
             << " the following values:"  << endl;
        for (int i = 0; i < count; i ++)   //(v)                     (1)
           cout << p1[i] << endl;

        delete [] p1;                    //vi)                       (1)
        return 0;
    }
```

Output:

```
please enter a value: 6
The elements in the array that p1 is pointing to have the following
values:
0
1
2
3
4
5

Process returned 0 (0x0)   execution time : 2.880 s
Press any key to continue.
```

(n)

Program:

```
#include <iostream>
using namespace std;
int main()
{
    typedef int* IntArrayPtr;
    IntArrayPtr a;                                                    (1)
    int size, max;
    cout << " How many scores will be entered? ";
    cin >> size;

    a = new int[size];

    cout << "Please enter " << size << " scores: " <<endl;
    for (int i = 0; i < size; i ++)                                   (1)
        cin >> a[i];

    total = 0;
        for (int i = 0; i < size; i ++)                               (1)
        total = total + a[i];

    average = total/size;


    cout << endl << " The average score is " << average;              (1)

    delete [] a;                                                      (1)
    return 0;
}
```

Output:

```
How many scores will be entered? 5
Please enter 5 scores:
67
55
78
65
75

 The highest score is 78
Process returned 0 (0x0)   execution time : 16.990 s
Press any key to continue.
```