**Luyanda Mncube 59448873 COS1511 Assignment 3**

**Question 1**

#include <iostream>

using namespace std;

void getprize(int ticketnum, char gender)

{

float prizemoney;

if (gender == 'm' && ticketnum > 30000)

prizemoney = ticketnum/90;

else

if (gender == 'f' && ticketnum > 20000)

prizemoney = ticketnum/80;

//output prize money

cout << "Congratulations, you won a prize of: R" << prizemoney;

}

void getprize(int ticketnum, char gender, int age)

{

float prizemoney;

if (age <= 21 )

prizemoney = age \* 40;

else

if ( age > 21)

prizemoney = age \* 30;

cout << "Congratulations, you won a prize of: R" << prizemoney;

}

int main()

{

int ticketnum, age;

char gender;

//Input ticket number

cout << "Please enter your ticket number: \n";

cin >> ticketnum;

//Input gender

cout << "Please enter your gender ('m' or 'f'): \n";

cin >> gender;

//check divisible by 6 AND 7

if (ticketnum % 6 == 0 && ticketnum % 7 == 0)

{

cout << "Please enter your age: \n" ;

cin >> ticketnum;

getprize(ticketnum,gender,age);

}

else

//Check divisible by 100

if (ticketnum % 100 == 0)

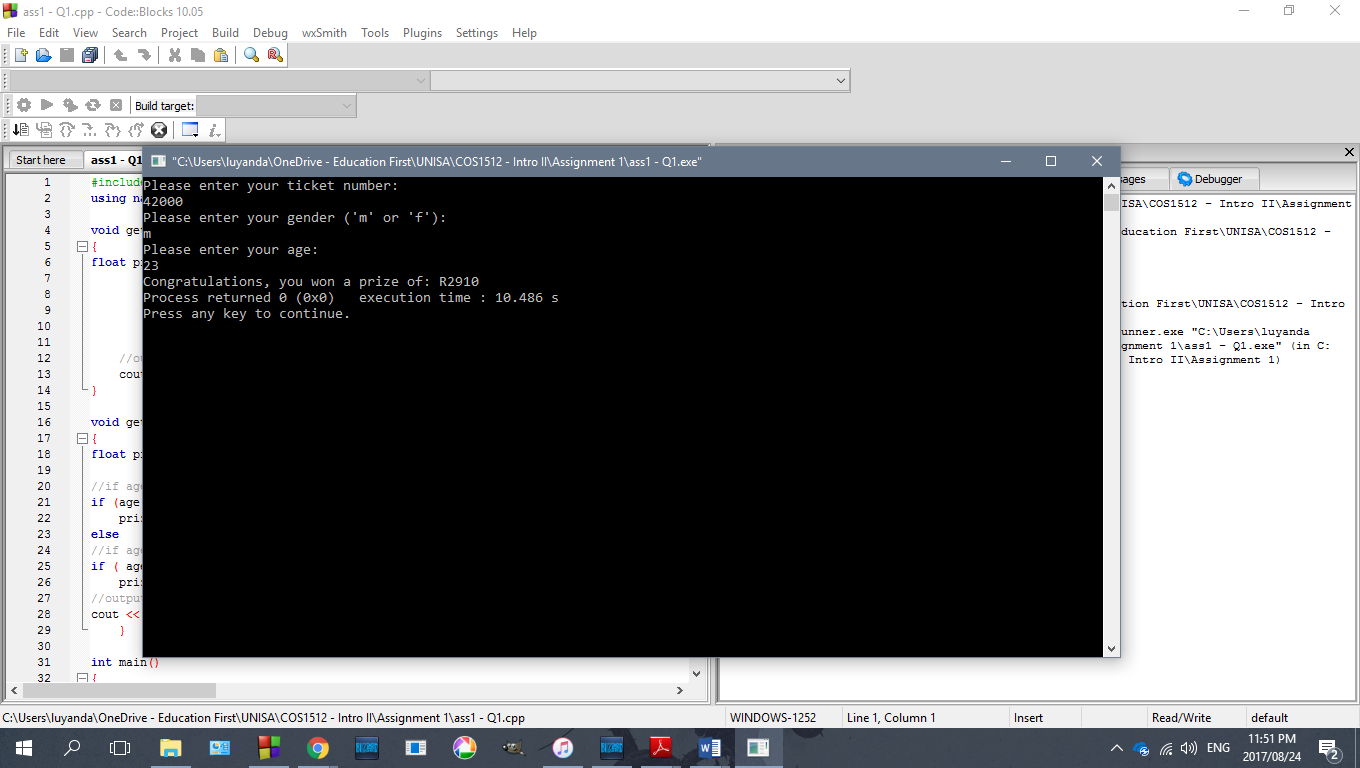
getprize(ticketnum,gender);

else

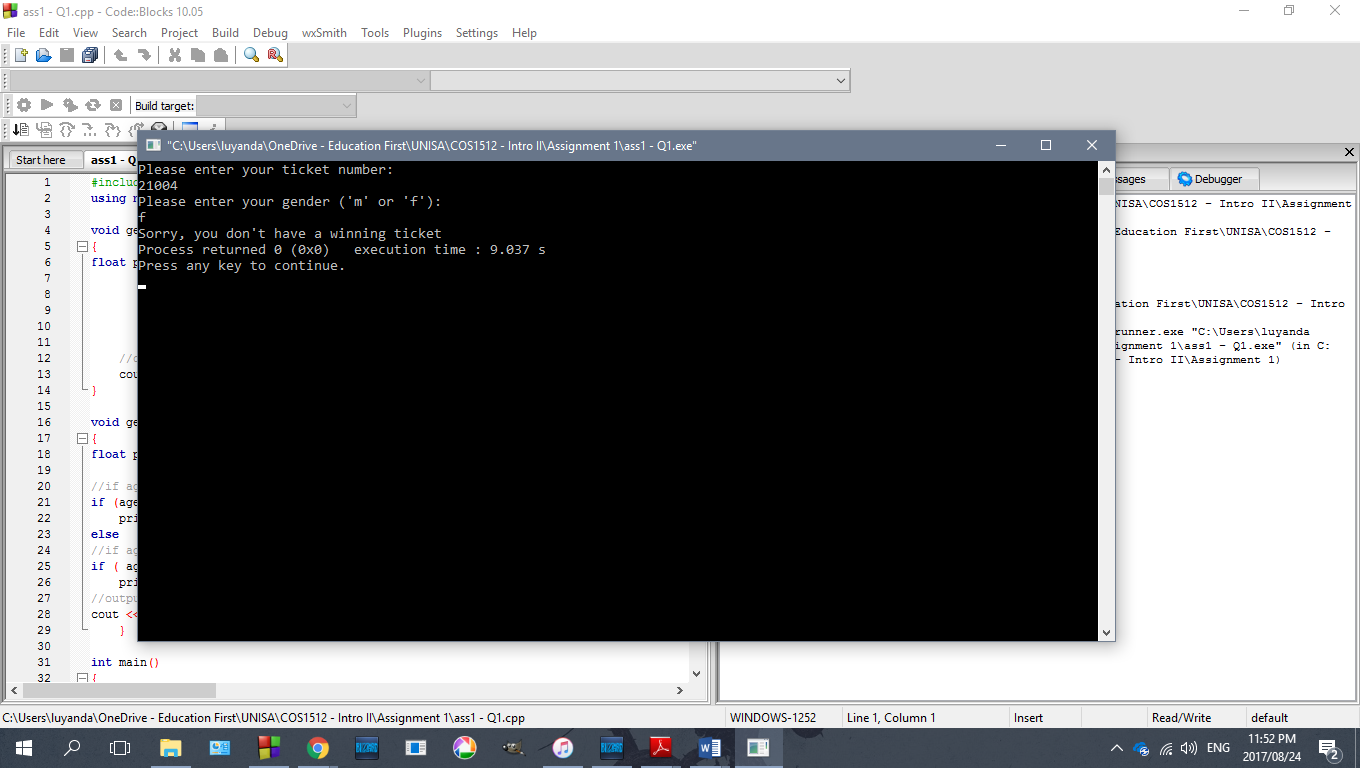
cout << "Sorry, you don't have a winning ticket" ;

return 0;

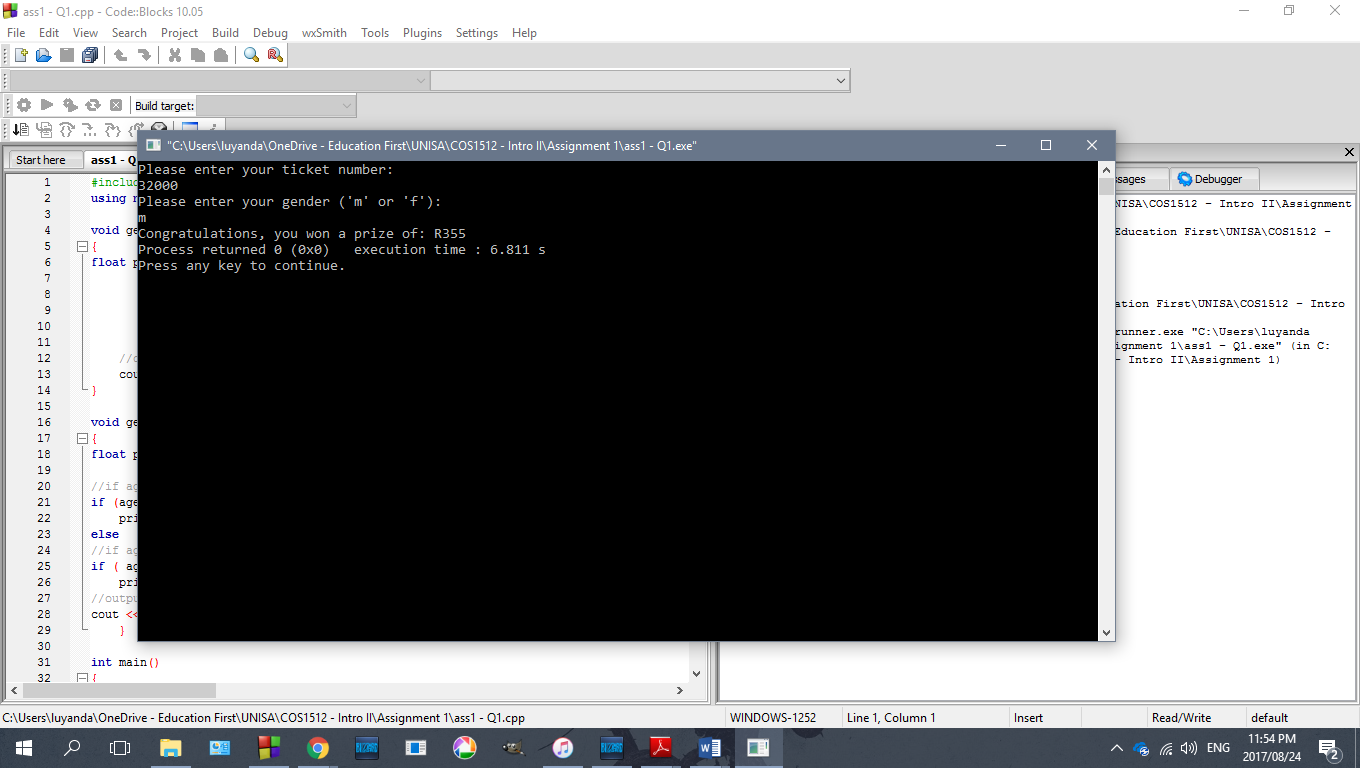
}



**Tested with input of 42000, a number divisible by 6 and 7 and by 100 (6\*7\*1000)**



**Tested with 21004, not divisible by 100 so not a winning ticket**



**Tested with 32000, divisible by 100 and anormal winning ticket**

**Question 2**

#include <iostream>

#include <cassert>

#include <string>

using namespace std;

void inputtime(int& itotal, int& iminutes, int& iseconds, int& ihours)

{

cout << "Please enter the time in the format (hh mm ss), with spaces \n";

cin >>ihours >> iminutes >> iseconds;

//assert macro

assert (iseconds <= 60);

assert(iminutes <= 60);

assert(ihours <= 24) ;

//Convert time to seconds

itotal = (ihours \* 3600) + (iminutes \* 60) + iseconds;

}

int main()

{

int imax = 0 , idifference = 0, itime2, sdiff, mdiff, hdiff,

itime1,iminutes,iseconds,ihours;

//Input and display first time

inputtime(itime1,iminutes,iseconds,ihours);

cout << "The first time inputted is \n" << ihours <<

":"<< iminutes << ":"<< iseconds;

cout << endl;

//Input and display second time

inputtime(itime2,iminutes,iseconds,ihours);

cout << "The second time inputted is \n" << ihours <<

":"<< iminutes << ":"<< iseconds;

cout << endl;

//Check which time is smaller

if (itime1 > itime2)

{

imax = itime1;

idifference = itime1 - itime2;

cout << "The first time entered is larger \n";

cout << endl;

}

else

{

imax = itime2;

idifference = itime2 - itime1;

cout << "The second time entered is larger \n";

cout << endl;

}

//Get time difference

mdiff = idifference / 60;

sdiff = idifference % 60;

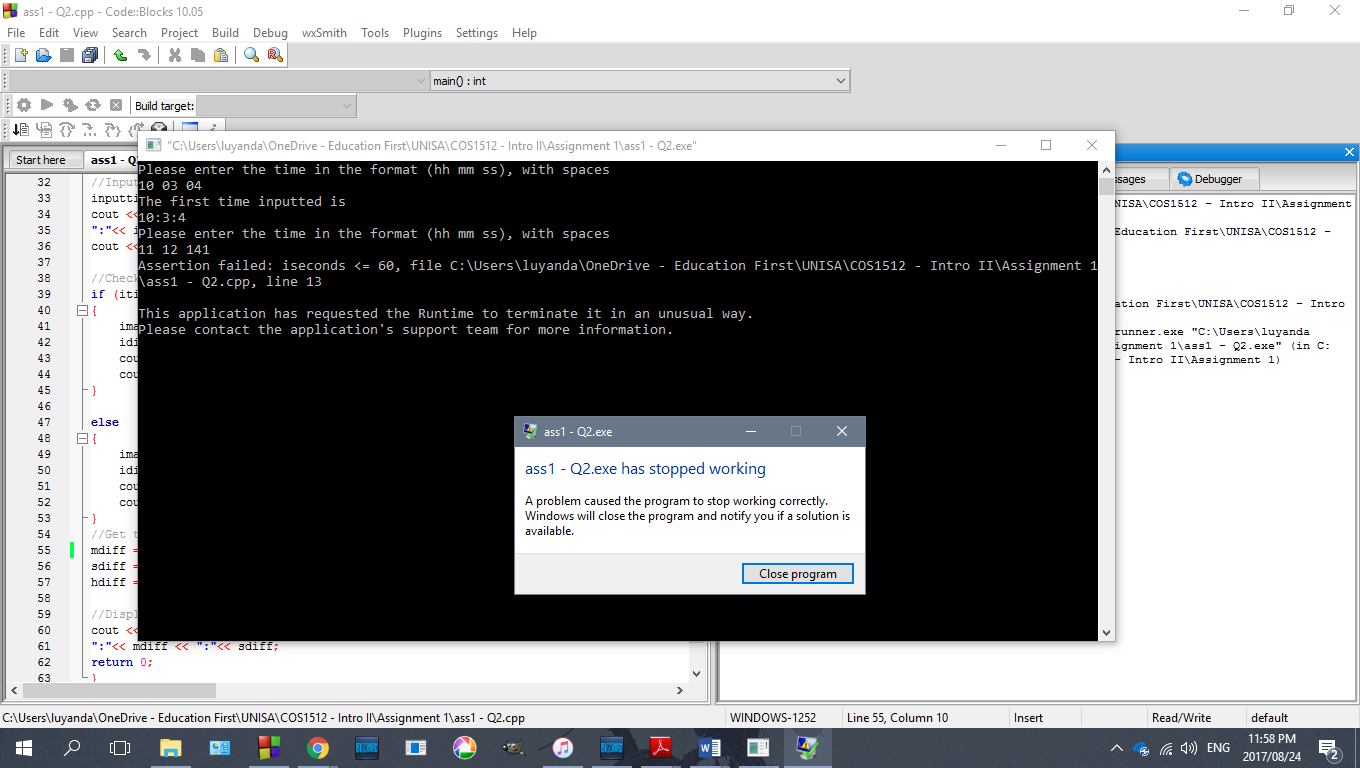
hdiff = mdiff / 60;

cout << "The time difference is " << hdiff <<

":"<< mdiff << ":"<< sdiff;

return 0;

}



**Assertion test**

**Question 3**

#include <cstdlib>

#include <iostream>

#include <fstream>

using namespace std;

int main()

{

double p[20], r;

int numElements = 0;

int medElement, middlepoint;

ifstream in\_stream;

in\_stream.open("file1.txt");

if(in\_stream.fail())

{

cout << "Failed to open a file" << endl;

system("PAUSE");

exit(0);

}

else

{

cout << "File open SUCCESSFUL. \n";

//Loop to count number of elements in text file

while(!in\_stream.eof())

{

in\_stream >> p[numElements];

numElements = numElements + 1;

}

}

in\_stream.close();

cout <<"Finding median... \n";

//If number of elements are EVEN

if (numElements%2 == 0)

{

cout << "The number of elements are EVEN \n";

middlepoint = numElements/2;

medElement = (p[middlepoint] + p[middlepoint+1])/2;

}

//If number of elements are ODD

else

{

cout << "The number of elements are ODD \n";

middlepoint = (numElements/2) + 1;

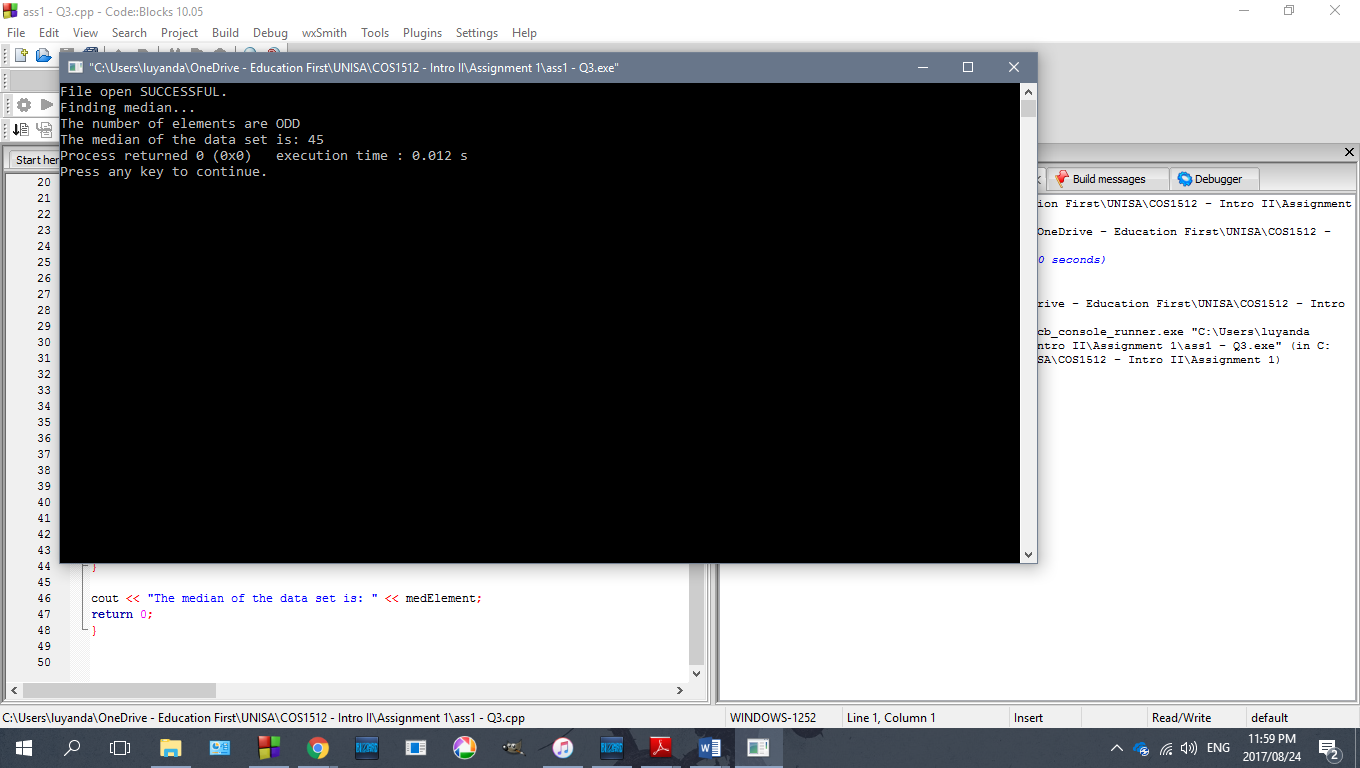
medElement = p[middlepoint];

}

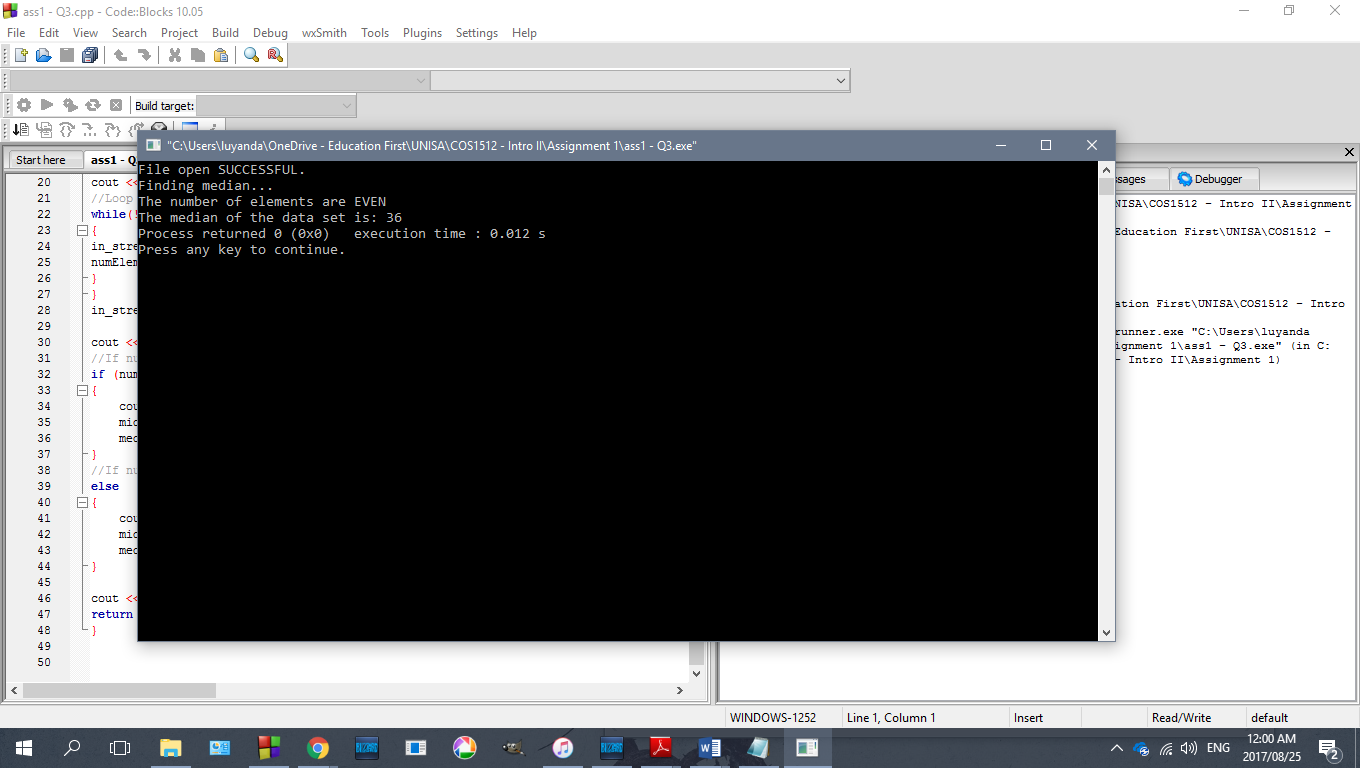
cout << "The median of the data set is: " << medElement;

return 0;

}



Test with first set of numbers



**Test with second set of numbers**

**Question 4 (Incomplete)**

#include <iostream>

#include <fstream>

#include <cstdlib>

using namespace std;

//create type definition pointer for dynamic array

typedef int\* p;

void encryptordecrypt (char file\_Read[16], char file\_Write[16], int option)

{

ifstream readtextfile;

int num\_characters=0, i=0;

char next;

//open input file stream

readtextfile.open(file\_READ);

//loop to count number of characters

while(!readtextfile.eof())

{

cout << ch;

readtextfile.get(ch);

num\_characters = num\_characters + 1;

}

//Create dynamic array p with numElements

charArrptr p;

charArrptr q;

p = new char [num\_characters];

q = new char [num\_characters]; //array to hold encryted characters

//loop to input characters from file into char array

in\_stream >> p[0];

while (! readtextfile.eof( ))

{

toupper(readtextfile.get(p[i])); //upper & lowercase letters

//encryption algorithm

if (option == 1)

{

if (p[i] == 'Z')

q[i] = 'A';

else

q[i] = p[i+1];

// cout ???? how to place into array/file

i++;

}

//decryption algorithm

else if (option == 2)

{

if (p[i] == 'A')

q[i] = 'Z';

else

q[i] = p[i-1];

// cout ???? how to place into array/file

i++;

}

}

}

int main()

{

int option;

char file\_name1[16], file\_name2[16];

//declare input & output streams

ifstream readtextfile;

ofstream writetextfile;

//Output user instructions into console

cout<<"Enter Your Option ";

cout<<"1. To Encrypt The File \n";

cout<<"2. To Decrypt The File ";

cout<<"Option : ";

cin>>option;

//Create file for encrypt or decrypt

cout << "Enter the file name to read/ write FROM (maximum of 15 characters):\n";

cin >> file\_READ;

cout << "Enter the file name to read/ write TO (maximum of 15 characters):\n";

cin >> file\_WRITE;

if(option == 1)

{

cout << "File name for READ is: " << file\_READ << endl;

//error check input stream

if(readtextfile.fail() = 0)

{

cout << "Failed to open read/input stream";

exit(1);

}

//Call encryptordecrypt function

}

else if (option == 2)

{

cout << "File name for decryption is: " << file\_name << endl;

//open output stream

writetextfile.open(file\_name);

//error check input stream

if(writetextfile.fail() = 0)

{

cout << "Failed to open write/output stream";

exit(1);

}

//Call decrypt option

encryptordecrypt (file\_READ[16],fileWRITE[16], option);

}

else

cout << "Please select valid option!";

return 0;

}

**Question 5 (incomplete)**

1. A pointer holds is the address in memory of a variable. A pointer itself is not a number but rather an area in memory which references a variable.
2. A dereferencing operator (\*) is used to dereferences a pointer. It is used with the abovementioned pointer to show the value of the variable the pointer is pointing to
3. p1 = p2, we are assigning the right hand side **pointer** to the left hand pointer using memory addresses.

In \*p1 = \*p2 we are assigning the **value** inside the right hand side variable to the left hand variable.

1. A dynamic variable is a variable that is created while a program is running. It is different from other variables which are allocated memory during compile time in general memory whereas dynamic are only allocated memory in the freestore during execution of a program.
2. New operator instantiates a new, nameless variable, with a specified data type. This is used to create dynamic variables as it actually holds a pointer and can only be accessed by referring to the pointer which the operator instantiates.
3. Delete is used so that a dynamic variable created will no longer take up the memory it was allocated in the freestore.
4. Freestore is a limited, designated area in memory that is reserved specifically for to be used for dynamic variables.
5. A dynamic variable is a variable that is created while a program is running . They have a reserved space in memory, and they are created and destroyed while the program is running.

Automatic variables are created when a function in which they are declared is called and are automatically destroyed when the function ends.

1. A dynamic array is one where the size is not specified when it is created. The size is only specified during runtime so hypothetically it could be of any size specified by the program (unlike in the case of normal arrays which have a set size specified at run time)
2. Dynamic arrays are useful as the size of the array can be changed or specified during the program execution. A user could for instance, input the size desired and change the size of the array during execution
3. Arrays and pointers are almost identical. An array declared as int a[10] is declaring a pointer to the first element in an array. In this sense, “a” is the pointer. The only difference between arrays and pointers is that “a” would be a constant in an array.

salary = 4500.00, increase = 475.00 double

vi. p1 = &a;

vii. delete p1;

(m)