Daren Sathasivam

ECC ID: 2558862

CSCI-2

Professor Ambrosio

***<u>Assignment #1 (60 points; due Friday, March 11, 2022 at 11:59 P.M.)</u>***

This project is designed to help you master pointers. To that end, you'll get the most out of it by working through the problems by hand. Only after that should you resort to running the programs (and stepping through them with the debugger) to check your understanding. Remember, on the final exam you'll have to be able to do problems like this by hand.

This "project" is more like a homework assignment, except for the last problem. There are six problems. The first five problems, you will submit a Word document or PDF in the format:

FirstInitialLastName-CS2-Section#-Assignment#.docx

For example, it'll look similar to this:

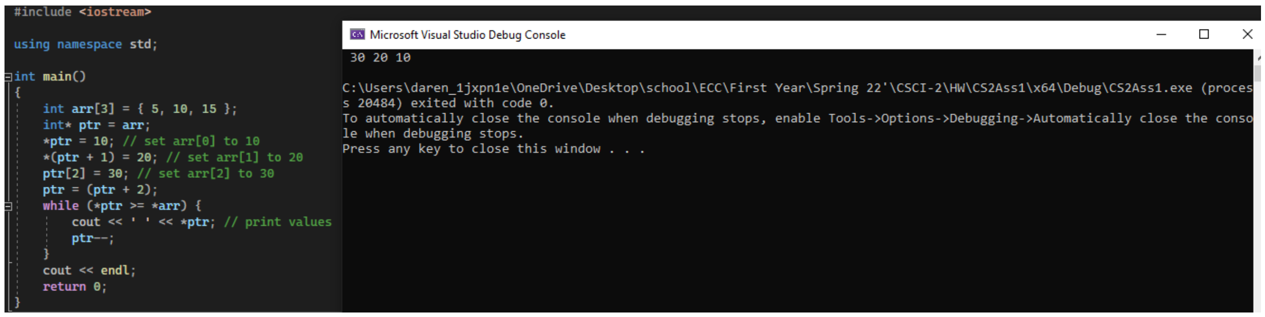
EAmbrosio-CS2-120-A1.docx

In problems that ask you to change code, make the few changes necessary to fix the code without changing its overall approach. For example, don't fix the program in problem 1a by changing it to

```
       int main()
       {
        cout << " 30 20 10" << endl;
 return 0;
       }
```

1. The subparts to this problem involve errors in the use of pointers. a. This program is supposed to write **`30 20 10`**, but it doesn't. Find all of the bugs and show a fixed version of the program:

```
int main()
 {
 int arr[3] = { 5, 10, 15 };
 int* ptr = arr;

 *ptr = 10; // set arr[0] to 10
 *(ptr + 1) = 20; // set arr[1] to 20
 ptr[2] = 30; // set arr[2] to 30
 ptr = (ptr + 2);
 while (*ptr >= *arr) {
 cout << ' ' << *ptr; // print values
 ptr--;
 }
 cout << endl;
 return 0;
}
```



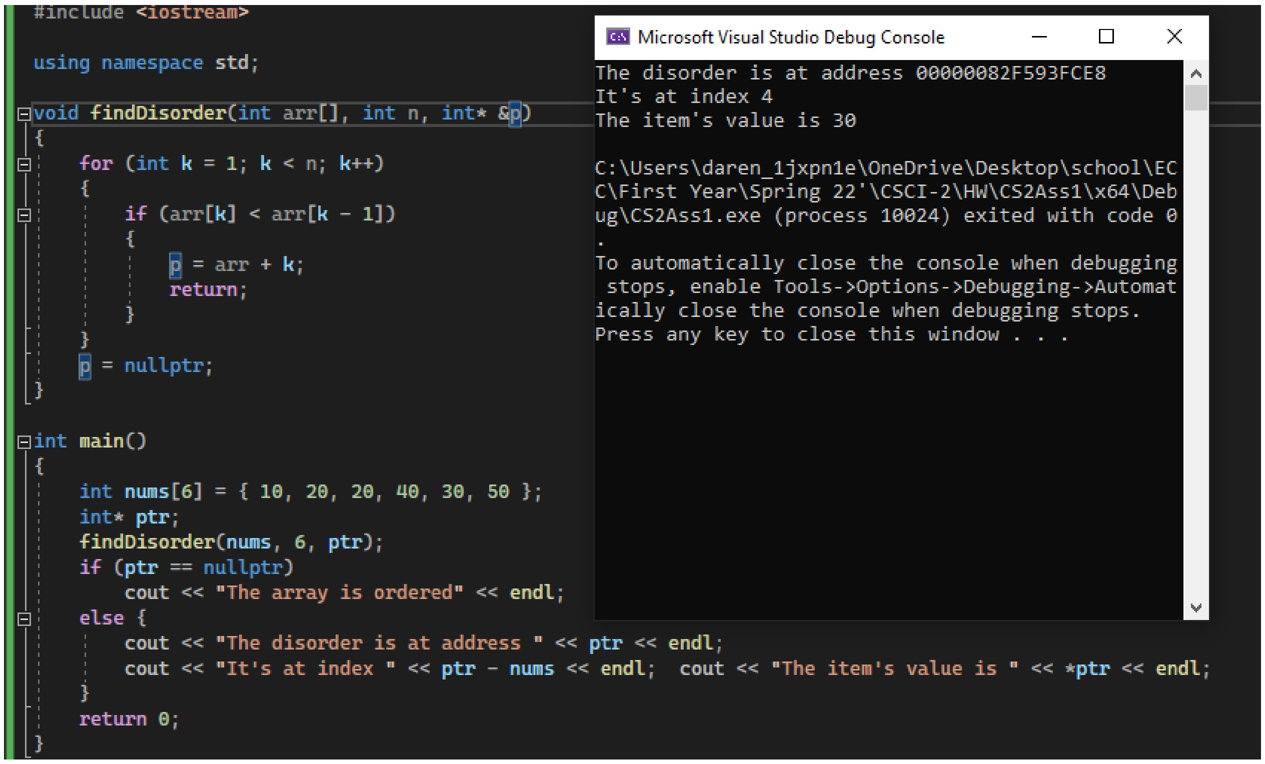
```
#include <iostream>

using namespace std;

int main()
{
    int arr[3] = { 5, 10, 15 };
    int* ptr = arr;
    *ptr = 10; // set arr[0] to 10
    *(ptr + 1) = 20; // set arr[1] to 20
    ptr[2] = 30; // set arr[2] to 30
    ptr = (ptr + 2);
    while (*ptr >= *arr) {
        cout << ' ' << *ptr; // print values
        ptr--;
    }
    cout << endl;
    return 0;
}
```

```
Microsoft Visual Studio Debug Console
 30 20 10

C:\Users\daren_1jxpn1e\OneDrive\Desktop\school\ECC\First Year\Spring 22'\CSCI-2\HW\CS2Ass1\x64\Debug\CS2Ass1.exe (proces
s 20484) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

b. The findDisorder function is supposed to find the first item in an array that is less than the element preceding it, and set the p parameter to point to that item, so the caller can know the location of that item. Explain why this function won't do that, and show how to fix it. Your fix must be to the function only; you must not change the the main routine below in any way, yet as a result of your fixing the function, the main routine below must work correctly.

-The function doesn't access the address of the parameter which leads to the error. The ampersand within the function parameter allows the address to be accessed.

```
void findDisorder(int arr[], int n, int* &p)
 {
     for (int k = 1; k < n; k++)
 {
      if (arr[k] < arr[k-1])
 {
      p = arr + k;
     return;
 }
 }
 p = nullptr;
 }

 int main()
 {
      int nums[6] = { 10, 20, 20, 40, 30, 50 };
     int* ptr;

 findDisorder(nums, 6, ptr);
      if (ptr == nullptr)
      cout << "The array is ordered" << endl;
      else {
 cout << "The disorder is at address " << ptr  << endl;
 cout << "It's at index " << ptr - nums << endl;  cout <<
"The item's value is " << *ptr << endl;  }
 return 0;
 }
```



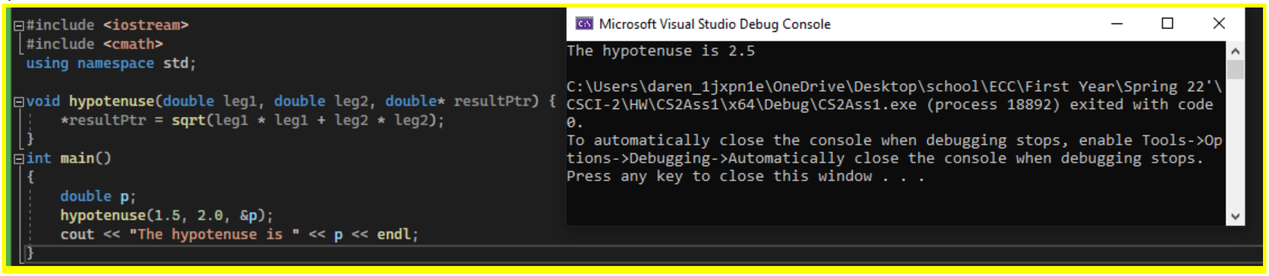
```
#include <iostream>

using namespace std;

void findDisorder(int arr[], int n, int* &p)
{
    for (int k = 1; k < n; k++)
    {
        if (arr[k] < arr[k - 1])
        {
            p = arr + k;
            return;
        }
    }
    p = nullptr;
}

int main()
{
    int nums[6] = { 10, 20, 20, 40, 30, 50 };
    int* ptr;
    findDisorder(nums, 6, ptr);
    if (ptr == nullptr)
        cout << "The array is ordered" << endl;
    else {
        cout << "The disorder is at address " << ptr << endl;
        cout << "It's at index " << ptr - nums << endl;  cout << "The item's value is " << *ptr << endl;
    }
    return 0;
}
```

```
The disorder is at address 00000082F593FCE8
It's at index 4
The item's value is 30

C:\Users\daren_1jxpn1e\OneDrive\Desktop\school\EC
C\First Year\Spring 22'\CSCI-2\HW\CS2Ass1\x64\Deb
ug\CS2Ass1.exe (process 10024) exited with code 0
.
To automatically close the console when debugging
 stops, enable Tools->Options->Debugging->Automat
ically close the console when debugging stops.
Press any key to close this window . . .
```

c. The hypotenuse function is correct, but the main function has a problem. Explain why it may not work, and show a way to fix it. Your fix must be to the main function only; you must not change the hypotenuse function in any way.

-The hypotenuse function stores the hypotenuse value under resultPtr. Instead of passing the ptr variable, pass the address of the variable when the function is called into the main. Then, the hypotenuse value will be stored at the address passed through the function and will output that value when called without the pointer variable.

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

 void hypotenuse(double leg1, double leg2, double* resultPtr)

 {
      *resultPtr = sqrt(leg1*leg1 + leg2*leg2);
}

 int main()
 {
     double p; //remove *
     hypotenuse(1.5, 2.0, &p);
     cout << "The hypotenuse is " << p << endl;
                                        //remove *
}
```



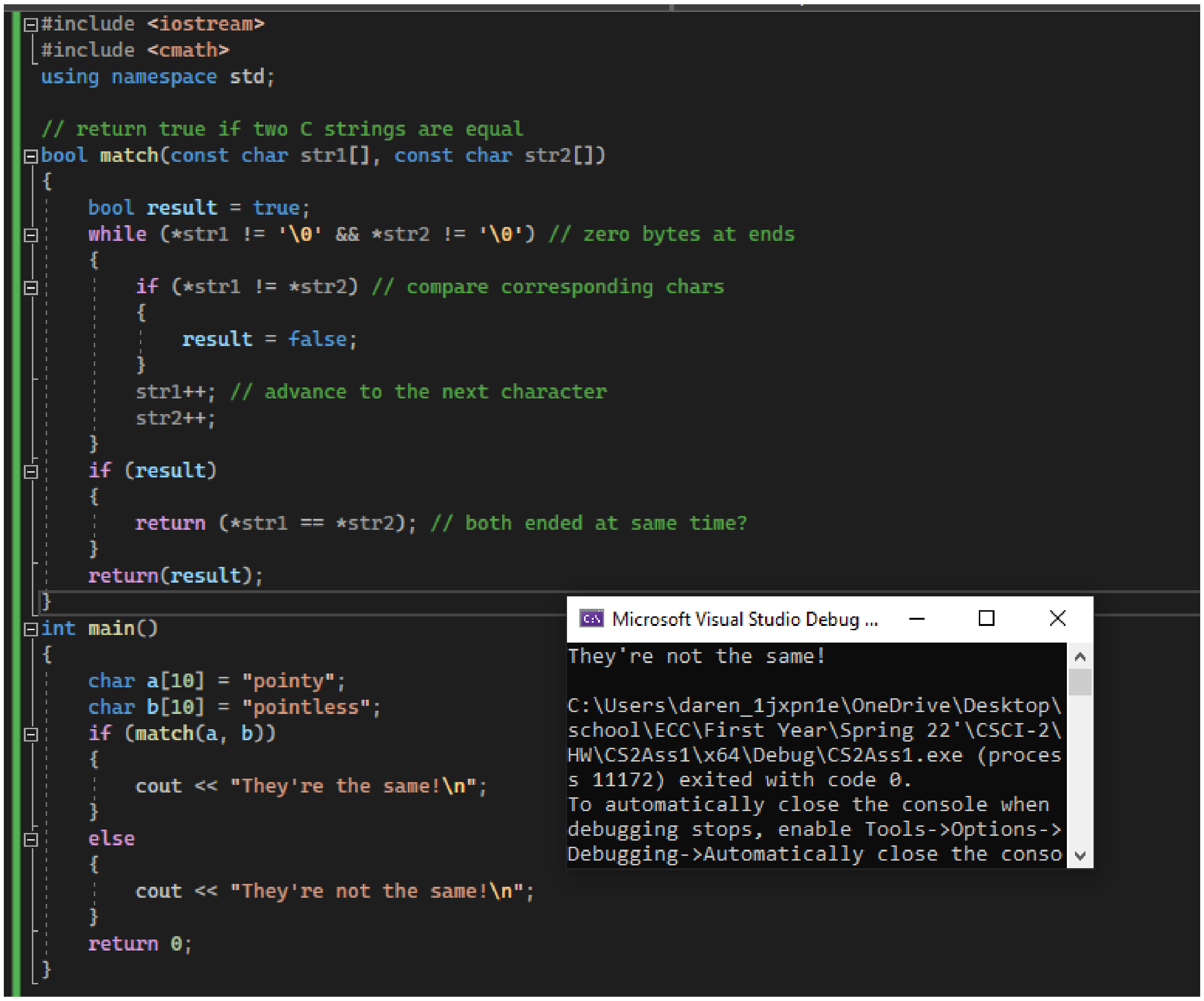
```
#include <iostream>
#include <cmath>
using namespace std;

void hypotenuse(double leg1, double leg2, double* resultPtr) {
    *resultPtr = sqrt(leg1 * leg1 + leg2 * leg2);
}
int main()
{
    double p;
    hypotenuse(1.5, 2.0, &p);
    cout << "The hypotenuse is " << p << endl;
}
```

```
The hypotenuse is 2.5

C:\Users\daren_1jxpn1e\OneDrive\Desktop\school\ECC\First Year\Spring 22'\
CSCI-2\HW\CS2Ass1\x64\Debug\CS2Ass1.exe (process 18892) exited with code
0.
To automatically close the console when debugging stops, enable Tools->Op
tions->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

d. The match function is supposed to return true if and only if its two C string arguments have exactly same text. Explain what the problems with the implementation of the function are, and show a way to fix them.

-The end of C-strings are marked with \0. So 0 and \0 are different. The return function is also not set to either true/false and only compares the addresses, not the end character. The str var should also be a pointer var so the value can be stored and accessed later on. The function should check whether or not the end is different and bool should be initialized to true unless the end char is different. Also needs an else-statement at the end to output “not the same” if the end chars are different.

```
 // return true if two C strings are equal
 bool match(const char str1[], const char str2[])
{
bool result = true;
while (*str1 != '\0' && *str2 != '\0') // zero bytes at ends
{
     if (*str1 != *str2) // compare corresponding chars
     {
          result = false;
     }
     str1++; // advance to the next character
     str2++;
 }
if(result)
{
 return (*str1 == *str2); // both ended at same time?
}
return(result);
}
int main()
{
 char a[10] = "pointy";
 char b[10] = "pointless";

 if (match(a,b))
 {
     cout << "They're the same!\n";
}
else
{
     cout << “They’re not the same!\n”;
}
 return 0;
}
```



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

// return true if two C strings are equal
bool match(const char str1[], const char str2[])
{
    bool result = true;
    while (*str1 != '\0' && *str2 != '\0') // zero bytes at ends
    {
        if (*str1 != *str2) // compare corresponding chars
        {
            result = false;
        }
        str1++; // advance to the next character
        str2++;
    }
    if (result)
    {
        return (*str1 == *str2); // both ended at same time?
    }
    return(result);
}
int main()
{
    char a[10] = "pointy";
    char b[10] = "pointless";
    if (match(a, b))
    {
        cout << "They're the same!\n";
    }
    else
    {
        cout << "They're not the same!\n";
    }
    return 0;
}
```

Microsoft Visual Studio Debug ...

```
They're not the same!

C:\Users\daren_1jxpn1e\OneDrive\Desktop\
school\ECC\First Year\Spring 22'\CSCI-2\
HW\CS2Ass1\x64\Debug\CS2Ass1.exe (proces
s 11172) exited with code 0.
To automatically close the console when
debugging stops, enable Tools->Options->
Debugging->Automatically close the conso
```

e. This program is supposed to write 1 4 9 16 25 36 49 64 81 100, but it probably does not. What is the program doing that is incorrect? (We're not asking you explain why the incorrect action leads to the particular outcome it does and we're not asking you to propose a fix to the problem.)

-The program's issues lies within when the function tries to access the array, it is a local variable whose memory is erased when the function is called over. The program does not allocate memory to store the values stored within the array.

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

int* computeSquares(int& n)
{
int *arr=new int[10];
n = 10;
for (int k = 0; k < n; k++)
arr[k] = (k+1) * (k+1);
return arr;
}

void f()
{
int junk[100];
for (int k = 0; k < 100; k++)
junk[k] = 123400000 + k;
}

int main()
{
int m;
int* ptr = computeSquares(m);
f();
for (int i = 0; i < m; i++)
cout << ptr[i] << ' ';
return 0;
}
```

2. For each of the following parts, write a single C++ statement that performs the indicated task. For each part, assume that all previous statements have been executed (e.g., when doing part e, assume the statements you wrote for parts a through d have been executed).

a. Declare a pointer variable named fp that can point to a variable of type string.

string *fp;

b. Declare fish to be a 5-element array of strings.

string fish[5];

c. Make the fp variable point to the last element of fish.

fp = &fish[4];

d. Make the string pointed to by fp equal to "tilapia", using the * operator.

*fp = “tilapia”;

e. Without using the fp pointer, and without using square brackets, set the fourth element (i.e., the one at index 3) of the fish array to have the value "salmon".

*(fish + 3) = “salmon”;

f. Move the fp pointer back by three strings.

fp = (fp-3);

g. Using square brackets, but without using the name fish, set the third element (i.e., the one at index 2) of the fish array to have the value "tuna".

fp[1] = “tuna”;

h. Without using the * operator, but using square brackets, set the string pointed to by fp to have the value "ono".

fp[0] = “ono”;

i. Using the == operator in the initialization expression, declare a bool variable named d and initialize it with an expression that evaluates to true if fp points to the string at the start of the fish array, and to false otherwise.

bool d = (fp == fish);

j. Using the * operator in the initialization expression, but no square brackets, declare a bool variable named b and initialize it to true if the string pointed to by fp is equal to the string immediately following the string pointed to by fp, and false otherwise.

bool b = (*fp == *(fp + 1));

3.

a. Rewrite the following function so that it returns the same result, but does not increment the variable ptr. Your new program must not use any square brackets, but must use an integer variable to visit each double in the array. You may eliminate any unneeded variable.

```
double computeAverage(const double* scores, int nScores)

{
 const double* ptr = scores;
 double tot = 0;
 int i = 0; //initialize int i var before for loop
 for (i = 0; i < nScores; i++)
     {
             tot += *(ptr + i); //delete incremented and use int i
     }                          //var to visit each double in array
      return tot/nScores;
 }
```

b. Rewrite the following function so that it does not use any square brackets (not even in the parameter declarations) but does use the integer variable k. Do not use any of the <cstring> functions such as strlen, strcpy, etc.

```
 // This function searches through str for the character chr.
// If the chr is found, it returns a pointer into str where  //
the character was first found, otherwise nullptr (not   //
found).

 const char* findTheChar(const char *str, char chr)

{                    //replace brackets with pointer

 for (int k = 0; (*(str + k)) != '\0'; k++)

{         //int k steps through array
 if ((*(str + k)) == chr) //replace brackets with pointer
      return (str+k);
}
return nullptr;
 }
```

c. Now rewrite the function shown in part b so that it uses neither square brackets nor any integer variables. Your new function must not use any local variables other than the parameters. (incremented/no brackets or int vars to step through)

```
const char* findTheChar(const char* str, char chr)

{

    while (*str != '\0')            //while loop with if statement to check array

    {                               //-> means no brackets/int var needed

            if (*str == chr)

            {

                    return (str);
```

```
                str++;   //incremented similar to part a
        }
    }
    return nullptr;
}
```

4. What does the following program print and why? Be sure to explain why each line of output prints the way it does to get full credit.

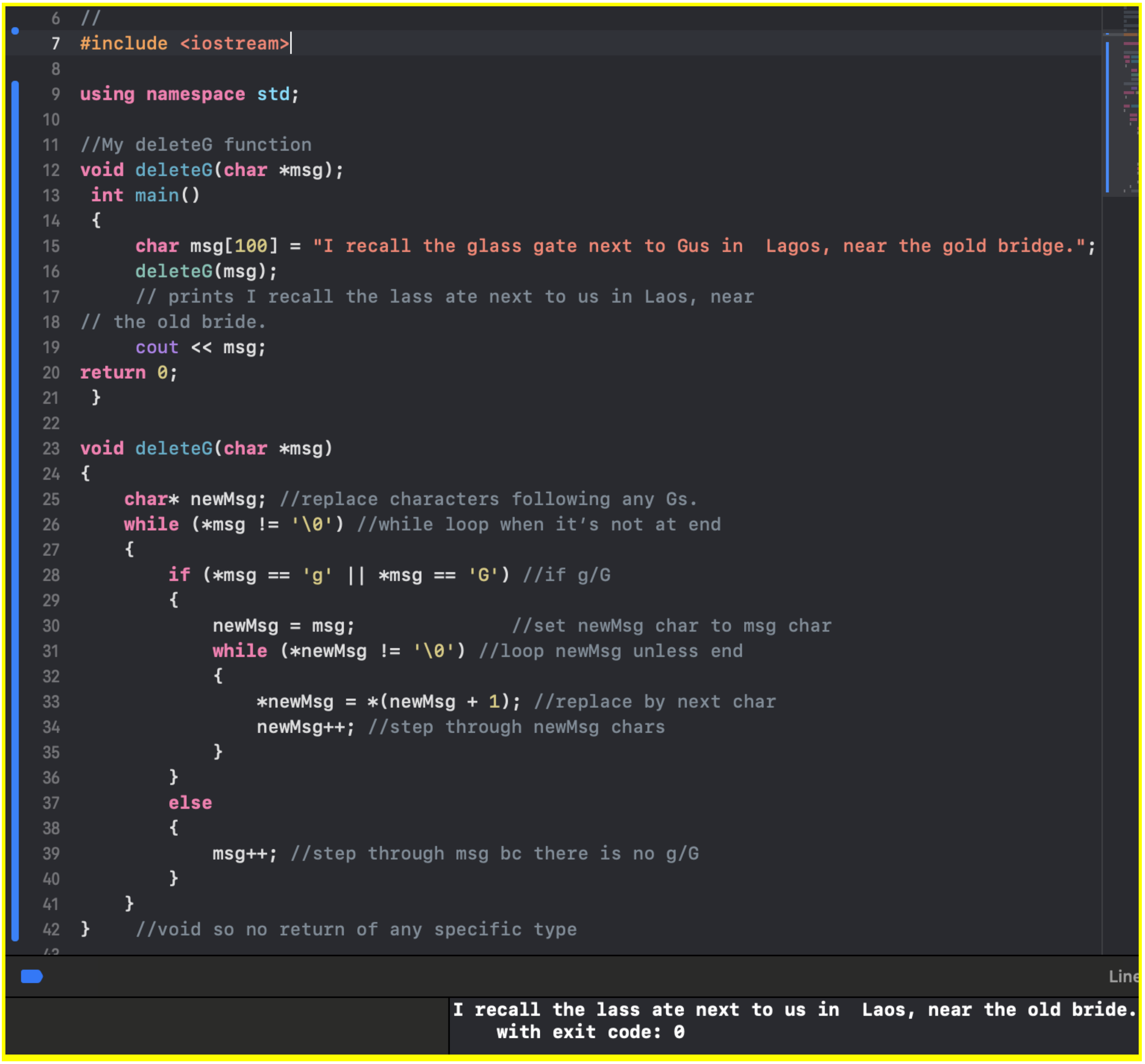
```
#include <iostream>
using namespace std;

int* minimart(int* a, int* b) //parameters that take in pointer var of a and b
{
     if (*a < *b)
     {
          return a;
     }
     else
     {
          return b;
     }
}

void swap1(int* a, int *b) //parameters that take in pointer var of a and b
{
    int* temp = a;
     a = b;
     b = temp; //swaps value in local so does not affect values outside of this function. Values within the array in the main remain the same.
}


void swap2(int* a, int *b) //parameters that take in pointer var of a and b
{
     int temp = *a;
     *a = *b;
     *b = temp;    //swap by pointer reference.
```

```
 }

 int main()
 {
     int array[6] = { 5, 3, 4, 17, 22, 19 }; //stores value into
an array with 6 different values. array[0]=5 … array[5]=19

      int* ptr = minimart(array, &array[2]); //calls minimart
function and takes values stored at array[0] and array[2]

     //returns value stored at *b, so returns 4.
      ptr[1] = 9; //ptr[1] goes from 3 to 9
      ptr += 2;
      *ptr = -1; //ptr+= is ptr[2] which goes from 4 to -1
      *(array+1) = 79; //store value of 79 in array[1]
//swaps values so array[0]=5, [1]=79, [2]=4. From [3], the values
are changed according to what is stored in the ptr.
array[3]=ptr[0]=4, [4]=ptr[1]=9, [5]=ptr[2]=-1.
//because ptr takes the address of array[2], then proceeds to the
next address's value. ptr[0]=4(previously), address of array[2].
ptr[1]=9(address of array[3], ptr[2]=-1(address of array[4].
Array becomes like this: array[0]=5, [1]=79, [2]=4, [3]=9,
[4]=-1, [5]=19.

      cout << "diff=" << &array[5] - ptr << endl; //Outputs
Address of value subtracted by the value of 9 (stored in pointer)

      swap1(&array[0], &array[1]); //calls swap1 function that
only alters the local vars. Main vars unchanged.
      swap2(array, &array[2]); //calls swap2 function that swaps
by pointer reference. Now, array[0]=4 and array[2]=5. The rest
remain the same.

      for (int i = 0; i < 6; i++)

      {
          cout << array[i] << endl; //will output 4, 79, 5, 9,
-1, and 19 on separate lines.
      }
 return 0;
 }
```

5. Write a function named deleteG that accepts one character pointer as a parameter and returns no value. The parameter is a C string. This function must remove all of the upper and lower case 'g' letters from the string. The resulting string must be a valid C string.

Your function must declare no more than one local variable in addition to the parameter; that additional variable must be of a pointer type. Your function must not use any square brackets and must not use the strlen or strcpy library functions.

```
//My deleteG function

void deleteG(char *msg);

 int main()
  {
      char msg[100] = "I recall the glass gate next to Gus in
Lagos, near the gold bridge.";
      deleteG(msg);
      // prints I recall the lass ate next to us in Laos, near
      // the old bride.
      cout << msg;
      return 0;
  }

void deleteG(char *msg)
{
      char* newMsg; //replace characters following any Gs.
      while (*msg != '\0') //while loop when it's not at end
      {
            if (*msg == 'g' || *msg == 'G') //if g/G
            {
                  newMsg = msg;               //set newMsg char to
msg char
                  while (*newMsg != '\0') //loop newMsg unless end
                  {
                        *newMsg = *(newMsg + 1) //replace by next
char
                        newMsg++; //step through newMsg chars
                  }
            }
            else
            {
                  msg++; //step through msg bc there is no g/G
            }
      }
}     //void so no return of any specific type
```



```cpp
//
#include <iostream>

using namespace std;

//My deleteG function
void deleteG(char *msg);
 int main()
 {
     char msg[100] = "I recall the glass gate next to Gus in  Lagos, near the gold bridge.";
     deleteG(msg);
     // prints I recall the lass ate next to us in Laos, near
// the old bride.
     cout << msg;
return 0;
 }

void deleteG(char *msg)
{
    char* newMsg; //replace characters following any Gs.
    while (*msg != '\0') //while loop when it's not at end
    {
        if (*msg == 'g' || *msg == 'G') //if g/G
        {
            newMsg = msg;              //set newMsg char to msg char
            while (*newMsg != '\0') //loop newMsg unless end
            {
                *newMsg = *(newMsg + 1); //replace by next char
                newMsg++; //step through newMsg chars
            }
        }
        else
        {
            msg++; //step through msg bc there is no g/G
        }
    }
}    //void so no return of any specific type
```

```
I recall the lass ate next to us in  Laos, near the old bride.
    with exit code: 0
```

6. For this problem, you will need to submit a C++ file. Make sure and include the following comments at the top (do this on all homework assignments from now on – this is required):

// Your Name
// CS 2, Section #122 (or 123, depending which one you're in)
// Assignment #, Problem #
// Summary of the program

Then, within the program, you will add pseudocode as appropriate to describe the steps of the program. This is in order to get in the habit of writing pseudocode and documenting your code. PSEUDOCODE IS REQUIRED FOR ALL PROGRAMS. Your C++ files should be in this format:

FirstInitialLastName-CS2-Section#-Assignment#-Problem#.cpp

For example, it'll look similar to this:

EAmbrosio-CS2-122-A1-P6.cppCSD

In this problem, you will write a program that can be used to gather statistical data about the number of movies college students see in a month. The program should perform the following steps:

a. Ask the user how many students were surveyed. An array of integers with that many students should then be dynamically allocated.
b. Allow the user to enter the number of movies each student saw into the array.
c. Calculate and display the average and median of the values entered. Write functions for calculating the average and the median.

Input Validation: Do not accept negative numbers for input. If they attempt to put in a negative number, re-prompt until at least 0 is entered.

**Turn It In:** Turn in the Word document or PDF and .cpp as one zip file in Canvas.