CS 31 Project 6 Homework

*For all questions the fix will be in blue.*

***Note for GitHub:*** *For this project we did not have an actual project per say but rather a series of homework problems to build mastery of pointers.*

**Question 1**

a) Bugs

\*ptr + 1 = 20; is wrong | should be \*(ptr + 1) = 20;

ptr—; is in wrong position | should be after cout << \*ptr << endl;

Additionally swap 30 and 10 so that the values will be printed out as 30, 20, 10

Fixed Version

int main()

{

int arr[3] = { 5, 10, 15 };

int\* ptr = arr;

\*ptr = **10**; // set arr[0] to 30

**\*(ptr + 1) = 20;**  // set arr[1] to 20

ptr += 2;

ptr[0] = **30**; // set arr[2] to 10

while (ptr >= arr)

{

cout << \*ptr << endl; // print values

**ptr--;**

}

}

b) This function will not work because the pointer must be passed by reference in order for the original pointer in the main routine to actually be modified. Otherwise the function will simply create a local pointer and then destroy it without modifying the original pointer. The fix is as follows:

void findMax(int arr[], int n, int\* **&**pToMax)

{

if (n <= 0)

return; // no items, no maximum!

pToMax = arr;

for (int i = 1; i < n; i++)

{

if (arr[i] > \*pToMax)

pToMax = arr + i;

}

}

int main()

{

int nums[4] = { 5, 3, 15, 6 };

int\* ptr;

findMax(nums, 4, ptr);

cout << "The maximum is at address " << ptr << endl;

cout << "It's at position " << ptr - nums << endl;

cout << "Its value is " << \*ptr << endl;

}

c) This program relies on uninitialized behavior; specifically, the pointer int\* ptr is never initialized within the main routine and yet when it gets passed into the computeCube function the value that ptr points to is somehow modified. This may not work all the time because it relies on unitialized behavior, and the fix is as follows:

void computeCube(int n, int\* ncubed)

{

\*ncubed = n \* n \* n;

}

int main()

{

**int temp = 0;**

int\* ptr **= &temp;**

computeCube(5, ptr);

cout << "Five cubed is " << \*ptr << endl;

}

d) The problem is that the array is passed as const to the function and yet the pointer pointing at the array is manipulated throughout the function. This is not allowed because a const pointer cannot be manipulated, so the parameter for the function should instead be another pointer which is then used to traverse the array and dereferenced whenever we are comparing the values of the original array. The fix is as follows:

// return true if two C strings are equal

bool strequal(**char\* str1, char\* str2**)

{

while (**\***str1 != ‘\**0’** && **\***str2 != ‘\**0’**)

{

if (**\***str1 != **\***str2) // compare corresponding characters

return false;

str1++; // advance to the next character

str2++;

}

return **\***str1 == **\***str2; // both ended at same time?

}

int main()

{

char a[15] = "Zhao";

char b[15] = "Zhou";

if (strequal(a,b))

cout << "They're the same person!\n";

}

e) The program creates a dangling pointer because it assigns ptr to the array created within the function getPtrToArray, so when that function is destroyed the array is also destroyed and ptr ends up pointing at random junk. Thus any attempts to modify ptr just leads to pointing at more random junk.

**Question 2**

a) double\* cat;

b) double mouse[5];

c) cat = &(mouse[4]);

d) \*cat = 25;

e) \*(mouse + 3) = 42;

f) cat -= 3;

g) cat[1] = 54;

h) cat[0] = 17;

i) bool b = (\*cat == cat[1]);

j) bool d = (cat == mouse);

**Question 3**

a) double mean(**double\* scores**, int numScores)

{

**int pos = 0;**

double tot = 0;

while (**pos < numScores**)

{

tot += **\*(scores + pos);**

**pos++;**

}

return tot/numScores;

}

b) // 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(**char\* str**, char chr)

{

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

if (**\*(str + k)** == chr)

return **(str + k)**;

return nullptr;

}

c) // 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(char\* str, char chr)

{

**while (\*str !=** ‘\**0’)**

**{**

if (**\*str** == chr)

return **str**;

**str++;**

**}**

return nullptr;

}

**Question 4**

The program will print as follows:

3

4

79

-1

9

22

19

3 // cout << &array[5] - ptr << endl; will print this out because ptr is pointing at the third element of the array and the distance between the sixth element of the array and the third element of the array is 3

4 // cout << array[i] << endl; within the for loop will print this out because the swap2 function switched the values of the first element of the array and the third element of the array

79 // cout << array[i] << endl; within the for loop will print this out because this line \*(array+1) = 79; changed the value of the second element of the array to 79

-1 // cout << array[i] << endl; within the for loop will print this out because this value was swapped to the third position from the first position and was originally changed while in the first position by this line \*ptr = -1;

9 // cout << array[i] << endl; within the for loop will print this out because these two lines changed the value of the fourth element ptr += 2;

ptr[1] = 9;

22 // cout << array[i] << endl; within the for loop will print this out because this was the value set during the initialization of the array

19 // cout << array[i] << endl; within the for loop will print this out because this was the value set during the initialization of the array

**Question 5**

**void removeS(char\* ptr)**

**{**

**while(\*ptr !=** ‘\**0’)**

**{**

**char\* temp = ptr;**

**if(\*ptr != 'S' && \*ptr != 's')**

**{**

**ptr++;**

**temp++;**

**}**

**else**

**{**

**while(\*temp == 'S' || \*temp == 's')**

**temp++;**

**\*ptr = \*temp;**

**if(\*temp !=** ‘\**0’)**

**\*temp = 's';**

**ptr++;**

**}**

**}**

**}**

int main()

{

char msg[50] = "She'll be a massless princess.";

removeS(msg);

cout << msg; // prints he'll be a male prince.

}