Ethan Wong

Professor Smallberg

305319001

20 November 2019

**Project 6**

1.

a) This program is supposed to write **30 20 10**, one per line. 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 = 30; // set arr[0] to 30

\*ptr + 1 = 20; // set arr[1] to 20 [incorrect syntax]

ptr += 2;

ptr[0] = 10; // set arr[2] to 10 [this doesn’t work correctly]

while (ptr >= arr) // [should be “<=” instead]

{

ptr--; // [should be ++}

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

}

}

**Corrected Version:**

int main()

{

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

int\* ptr = arr;

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

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

\*(ptr + 2) = 10; // set arr[2] to 10

while (ptr <= arr)

{

cout << \*ptr << endl; // print values: arr[0], arr[1], arr[2]

ptr++;

}

}

b) The problem with the findMax function is that the pointer “int\* pToMax” is passed by value. This means that in the code of the function, a copy of the pointer is being used, not the actual pointer. This causes the function to work incorrectly. In order to resolve this, the pointer must be changed to be passed by reference. The function header should read as follows:

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

c) The problem with the main function is that the pointer is declared but it never gets initialized. In other words, the pointer isn’t actually pointing at anything. This can be resolved by changing the main function to look like this:

int main()

{

int z = 5;

int\* ptr = &z;

computeCube(z, ptr);

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

}

d) The issue with this function is that it is comparing the memory location of the characters of the C strings, not the actual value of the characters of the C strings. Since the characters of the two different C strings can not share the same location in memory, the Boolean will always return false. To remedy this, the program can be changed to read as follows:

bool strequal(const char\* str1, const 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?

}

e) The problem with this program is that the “anArray” variable in the “getPtrToArray” is a local variable that only works within the scope of that function. This means that once the program finished with this function, the elements of “anArray” are gone and set to undefined values. This causes the program to run incorrectly because it will depend on undefined behavior.

2.

a) double\* cat;

b) double mouse[5];

c) cat = mouse + 4;

d) \*cat = 25;

e) \*(mouse + 3) = 54

f) cat -= 3;

g) cat[1] = 27;

h) cat[0] = 42;

i) bool b = (\*cat == \*(cat + 1));

j) bool d = (cat == &mouse[0]);

3.

a)

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

{

const double\* ptr = scores;

double tot = 0;

for (int x = 0; x < numScores; x++)

{

tot += \*(ptr + x);

}

return tot/numScores;

}

b)

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

{

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

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

return str + k;

return nullptr;

}

c)

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

{

for(; \*str!= 0, str++)

if (\*str == chr)

return str;

return nullptr;

}

4.

int\* maxwell(int\* a, int\* b) //returns pointer that points to the greater dereference value

{

if (\*a > \*b)

return a;

else

return b;

}

void swap1(int\* a, int\* b) //swaps the copies of the pointers passed into it -- doesn’t affect the ints that are being pointed at

{

int\* temp = a;

a = b;

b = temp;

}

void swap2(int\* a, int\* b) //swaps the values the pointers are pointing to

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main()

{

int array[6] = { 5, 3, 4, 17, 22, 19 }; //initializes an array of ints size 6

int\* ptr = maxwell(array, &array[2]); //initializes a pointer that points to &array[0] since value of array[0] > array[2]

\*ptr = -1; //sets the value at &array[0] to -1

ptr += 2; //moves the pointer to &array[2]

ptr[1] = 9; //sets the value of &array[3] to 9

\*(array+1) = 79; //sets the value of &array[1] to 79

cout << &array[5] - ptr << endl; //prints the difference of &array[5] and &array[2], which is 3 because the address locations of an array are sequential

swap1(&array[0], &array[1]); //swaps the addresses of &array[0] and &array[1]

swap2(array, &array[2]); //swaps the values of &array[0] and &array[2]

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

cout << array[i] << endl; //prints out each value of the array, which is now: 4, 79, -1, 9, 22, 19

}

The output is:

3

4

79

-1

9

22

19

5.

void removeS (char\* msg)

{

char\* result = msg;

for (; \*msg != 0; msg++)

{

if (\*msg == ‘s’ || \*msg == ‘S’)

continue;

}

else

{

\*result == \*msg;

result++;

}

\*result = 0;

}