Maeneka Grewal – 204901948

CS 31

Project 6

1a) The bugs are: \*ptr + 1 is not a valid statement and the while loop simply doesn’t work because the de-incrementing of ptr isn’t in the right place, and it also prints backwards, from the last element to the first element, and. Here is the fix:

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;

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

ptr -= 2;

while (ptr < arr + 3)

{

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

ptr++;

}

}

1b) The pointer is being passed by value, and not by reference. The function parameter should include a &\* next to the int, not just a \*. The fix is shown below.

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;

}

}

}

1c) The function computeCube is passed a pointer, and tries to store the cubed integer inside this pointer. However, the pointer does not point to anything. In main, the pointer ‘ptr’ is declared but does not point to anything, so there is no address to store the cubed integer in. The fix is to declare an integer in main that the pointer can point to. It is shown below.

int cubed = 0;

int\* ptr = &cubed;

computeCube(5, ptr);

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

1d) The line for the while loop condition contains an error, because it is trying to compare the pointer to an integer (0), and not the char the pointer is pointing. It is checking to see if the addresses of the pointers are equal to zero. Instead, the star operator should be used to follow the pointers to the char, and see if that char is equal to the null byte. Also, the comparison of the ‘if’ needs the star operator because once again, it is comparing an address and not the chars themselves. The same problem occurs in the last line with the return. The fix is shown below.

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?

}

1e) The ptr is not getting properly initialized, because getPtrtoArray is returning the address of an array that is only known to the function and not to main. So ptr is not properly pointing to anArray. In addition, the function f is useless but is still somehow messing up the program. I really can’t explain that, I just noticed that when you comment out the f(); line, the values printed change.

2a) double\* cat;

2b) double mouse[5];

2c) cat = &mouse[4];

2d) \*cat = 25;

2e) \*(mouse + 3) = 42;

2f) cat -= 3;

2g) cat[2] = 54;

2h) cat[0] = 27;

2i) bool b = (\*cat == \*(++cat));

2j) bool d = (\*cat == mouse[0]);

3a)

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

{

double tot = 0;

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

{

tot += scores[i];

}

return tot/numScores;

}

3b)

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

{

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

{

if (\*str == chr)

return str;

str++;

}

return nullptr;

}

3c)

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

{

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

if (\*str == chr)

return str;

return nullptr;

}

4a)

The pointer is first initialized to the element 0 of the array, because maxwell was passed element 0 of the array and element 2, and 5 is greater than 4. Then, element 0 is set to 3. The pointer is then set to point to element 2 of the array. Since it is now pointing at element 2, the ptr[1] = 9 statement makes element 3 equal 9. Then, element 1 is set to 79. Therefore, the **first cout statement prints 3**, since position 5 minus position 2 is 3.

Then, the first swap statement switches element 0 and 1. So element 0 is now 79 and element 1 is -1. The second swap statement switches element 0 and element 2, so now element 0 is 4 and element 2 is 79. Therefore, **the second cout statement prints 4, -1, 79, 9, 22, 19 with a new line in between each number.**

5.

void removeS(char\* c)

{

char\* noS = c;

for (; \*noS != '\0'; c++)

{

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

{

\*noS = \*c;

noS++;

}

}

c = noS;

}