Rahul Reddy

CS 31

Professor Smallberg

Project 6

1)

a)

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 + 1 should be in parentheses**

**ptr += 2;**

**ptr[0] = 10;** // set arr[2] to 10 **//should be just \*ptr**

**while (ptr >= arr) //this would print out in reverse order**

**{**

**ptr--; //should be incrementing**

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

}

}

Fixed 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 + 2))

{

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

ptr++;

}

}

b)

Since this function is of void type, it returns nothing and when the 3rd argument, “int\* pToMax”, is changed in any way, then these changes will not be saved when the function returns nothing. To solve the issue, the only thing that needs to be changed is the third parameter of the function. Instead of “int\* pToMax”, it should be “int\* &pToMax”, so that it is passed by reference. Therefore, the function will be “void findMax(int arr[], int n, int\* &pToMax).”

c)

The pointer needs to be initialized to a memory address or there will be a compilation. In the correct version, there is an initialization of an int variable along with the pointer being set to the memory address of the variable.

int main()

{

**int\* ptr; //this should be initialized to something**

computeCube(5, ptr);

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

}

Correct Fix:

int main()

{

int z;

int\* ptr = &z;

computeCube(5, ptr);

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

}

d)

// return true if two C strings are equal

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?

}

Correct Fix:

// return true if two C strings are equal

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 program is failing because anArray is a variable declared locally within an int function , so every time this function is called, the array and its values will be cleared. The pointer variable will now point to a memory address that that no longer has any value and will cause undefined behavior. Another thing that could happen is that other arbitrary values will end up in the memory address from the junk values in the array. This will cause random values for the pointer to point at.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2)

a. double\* cat;

b. double mouse[5];

c. cat = mouse + 4;

d. \*cat = 25;

e. \*(mouse + 3) = 54;

f. cat = cat – 3;

g. cat[1] = 42;

h. cat[0] = 27;

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;

int k = 0;

while (k < numScores)

{

tot = tot + \*(ptr + k);

k++;

}

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)

{

while(\*str != ‘\0’) {

if (\*str == chr){

return str;

}

str++;

}

return nullptr;

}

4)

#include <iostream>

using namespace std;

int\* maxwell(int\* a, int\* b)

{

if (\*a > \*b)

return a;

else

return b;

}

void swap1(int\* a, int\* b)

{

int\* temp = a;

a = b;

b = temp;

}

void swap2(int\* a, int\* b)

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main()

{

int array[6] = { 5, 3, 4, 17, 22, 19 };

int\* ptr = maxwell(array, &array[2]); //compares 5 and 4 and returns the memory address of whichever is greater

\*ptr = -1; //sets index 0 in the array to -1

ptr += 2; //sets the pointer pointing at arr[2]

ptr[1] = 9; //arr[3] = 9

\*(array+1) = 79; //arr[1] = 79

cout << &array[5] - ptr << endl; //prints 3 since that’s (array+5)-(array+2). This is how many elements are between the two values being pointed to

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

swap2(array, &array[2]); //swaps values, -1 and 4, which are index 0 and 2 respectively

for (int i = 0; i < 6; i++) //prints the array with each index on a newline

cout << array[i] << endl;

}

PRINTS:

3 (when \*(array+5 – ptr) = 3)

4 (switches with -1)

79 (when (\*array+1) = 79)

-1 (switches with 4)

9 (when ptr[1] = 9)

22 (stays the same)

19 (stays the same)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5)

void removeS(char\* ptr) {

char\* ptr1 = ptr;

while (\*ptr1 != ‘\0’) {

if (\*ptr1 == ‘s’ || \*ptr1 == ‘S’) {

++ptr1;

}

else {

\*ptr = \*ptr1;

++ptr;

++ptr1;

}

}

\*ptr = ‘\0’;

}