Project 6

1a)

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 -= 3;

while (ptr < &arr[2])

{

ptr++;

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

}

}

1b) In this program, you’re declaring a pointer in the main function and passing it as an argument to findMax, while the function findMax declares a different pointer pToMax as a parameter and updates where it is pointing. While pToMax does point to the max item, it doesn’t update ptr at all. In other words, ptr isn’t modified when pToMax is.

**Fix:**

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;

}

1c) The pointer ptr declared in the main function is not initialized, so it operates as an uninitialized pointer, which is undefined behavior.

**Fix:**

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int n;

int\* ptr = &n;

computeCube(5, ptr);

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

}

1d) The function strequal tries to compare two C strings with “str1 == str2” and increment through each C string with str1++ and str2++, respectively, which simply isn’t a legitimate method of traversing through and comparing arrays/C strings. Also, strequal will always advance to the next character of *both* C strings before re-checking the condition for the while loop, meaning that any method of comparison checking the position within each C string after the while loop ends will be ineffective, as the positions will always come out to be the same. In other words, the strings will always appear to have ended at the same time.

**Fix:**

// return true if two C strings are equal

bool strequal(const char str1[], const char str2[])

{

const char\* ptr1 = str1;

const char\* ptr2 = str2;

int count1 = 0;

int count2 = 0;

while (\*ptr1 != '\0' && \*ptr2 != '\0')

{

if (\*ptr1 != \*ptr2) // compare corresponding characters

return false;

if (ptr1[1] != '\0')

count1++;

if (ptr2[1] != '\0')

count2++;

ptr1++; // advance to the next character

ptr2++;

}

return count1 == count2; // both ended at same time?

}

int main()

{

char a[15] = "Shi";

char b[15] = "Shi";

if (strequal(a,b))

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

}

1e) The function, in the statement: int\* ptr = getPtrToArray(n); , is initializing the pointer ptr with the memory location of the first element of the local variable anArray that the function getPtrToArray is returning. The program ends up erasing anArray after getPtrToArray is completed and leaves ptr pointing to a random, irrelevant memory address.

2a) double\* cat;

2b) double mouse[5];

2c) cat = &mouse[4];

2d) \*cat = 25;

2e) \*(mouse + 3) = 54;

2f) cat -= 3;

2g) cat[1] = 42;

2h) cat[0] = 27;

2i) bool b = (\*cat == \*(cat+1)) ? true : false;

2j) bool d = (cat == &mouse[0]) ? true : false;

3a)

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

{

const double\* ptr = scores;

double tot = 0;

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

{

tot += \*(ptr + i);

}

return tot/numScores;

}

3b)

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

{

int k = 0;

const char\* ptr = str;

while (\*(ptr+k) != '\0'){

if (\*(ptr+k) == chr){

return ptr+k;

}

k++;

}

return nullptr;

}

3c)

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

{

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

if (\*str == chr)

return str;

}

return nullptr;

}

4)

**Output:**

3

4

79

-1

9

22

19

**Explanation** (commented throughout the code in *italics*)**:**

#include <iostream>

using namespace std;

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

{

if (\*a > \*b) *// 5 > 4, so returns a pointer to 5 at pos 0 of array*

return a;

else

return b;

}

void swap1(int\* a, int\* b) *//doesn’t change the integer values of the array at all*

{

int\* temp = a;

a = b;

b = temp;

}

void swap2(int\* a, int\* b) *//swaps the integers at the memory locations that pointers //a and b are pointing to*

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main()

{

int array[6] = { 5, 3, 4, 17, 22, 19 }; *//initial array integer values*

int\* ptr = maxwell(array, &array[2]); *//ptr initially points to pos 0 of array (&array[0])*

\*ptr = -1; *//the integer ptr is pointing to at pos 0 of array is updated to -1*

ptr += 2; *//ptr moves forward two, so it’s now pointing to pos 2 of array*

ptr[1] = 9; *//the integer one to the right of the integer ptr is pointing to (aka the //integer at pos 3 of array) is updated to be 9, ptr still pointing to pos 2 of array*

\*(array+1) = 79; *//the integer at pos 1 of array (&array[1]) is updated to be 79*

*//so now the array is: { -1, 79, 4, 9, 22, 19 }*

cout << &array[5] - ptr << endl; ***//prints 3 and new line: at this point, ptr is currently at //position two of array, which equals &array[0+2] = &array[2], and &array[5] - &array[2] = 5 // - 2 = 3 (+endl means new line after the 3)***

swap1(&array[0], &array[1]); *//no integers in the array change value*

swap2(array, &array[2]); *//swaps the integers at pos 0 and 2 (-1 and 4 switch places)*

***//so now the array is: { 4, 79, -1, 9, 22, 19 }***

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

cout << array[i] << endl; ***//iterating through the entire array one-by-one, prints //out the integers at pos 0-5 of array, with a new line after each integer***

}

5)

void removeS(char\* str){

while (\*str != '\0'){

if (\*str == 's' || \*str == 'S'){

char\* ptr = str;

while (\*ptr != '\0'){

\*ptr = \*(ptr+1);

ptr++;

}

str--;

}

str++;

}

}