1.

1. Supposed to write 30, 20, 10 one per line.

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 **This is not the appropriate syntax**

ptr += 2;

**ptr[0] = 10;** // set arr[2] to 10  **No \*, so the pointer isn’t followed. Doesn’t**

**actually set ar[2] to 10.**

**while (ptr >= arr)**

{ **This while loop will print the array in reverse order (10,**

**ptr--; 20, 30)**

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;

ptr++;

}

}

1. The reason why the findMax function is failing is due to the fact that the pointer variable pToMax is passed by value, rather than passed by reference. Thus, a copy is used when pToMax is initialized and the actual value of pToMax outside of the function stays the same afterward. In order to fix this, simply add a “&” after the int\* pointer declaration in the function implementation like so:

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

1. The main function declares the integer pointer ptr, but doesn’t initialize it before passing it through the computeCube function. As the pointer was not initialized, it’s not pointing at anything in particular in the program, and thus will crash. To fix this, simply declare a variable before the pointer and set the pointer to the new variable’s address:

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int x;

int\* ptr = &x;

computeCube(5, ptr);

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

}

1. The problem with the strequal function lies in the fact that the function is comparing the arrays themselves, rather than the individual characters in the array. Since no information is given in square brackets, by saying “while (str1 != 0 && str2 != 0)”, the memory addresses of str1 and str2 are being compared and thus, we will always stay in the while loop. In order to fix this, alter the program to take in a pointer to str1 and str2. This will allow proper movement within the array to check characters

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?

}

1. The problem with this program lies in the second function, getPtrToArray. Within this function’s implementation, the integer array anArray[100] is made. At the end of the function, the pointer address to the beginning of anArray is returned. However, since anArray is a local array, it will be deleted once the function is done. Thus, the returned pointer will point towards garbage data rather than the numbers expected.

2.

1. double\* cat;
2. double mouse[5];
3. cat = mouse + 4;
4. \*cat = 25;
5. \*(mouse + 4) = 54;
6. cat -= 3;
7. cat[1] = 27;
8. cat[0] = 42;
9. bool b = (\*cat == \*(cat+1));
10. bool d = (cat == &mouse[0]);

3.

1. Rewritten function

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

{

const double\* ptr = scores;

double tot = 0;

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

{

tot += \*(ptr+x);

}

return tot/numScores;

}

1. Rewritten function

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;

}

1. Rewritten part b.

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

{

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

if (\*str == chr)

return str;

return nullptr;

}

4.

#include <iostream>

using namespace std;

int\* maxwell(int\* a, int\* b) **//This function returns a pointer to the value of a and b that is greater**

{

if (\*a > \*b)

return a;

else

return b;

}

void swap1(int\* a, int\* b) **//Doesn’t return anything. Swaps the addresses of a and b (not the actual values)**

{

int\* temp = a;

a = b;

b = temp;

}

void swap2(int\* a, int\* b) **//Doesn’t return anything.** **Swaps the actual values of what a and b are pointing to.**

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main()

{

int array[6] = { 5, 3, 4, 17, 22, 19 }; **//Creates an array 6 integer elements**

int\* ptr = maxwell(array, &array[2]); **//Creates a pointer to a array[0] has 5 > 4**

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

ptr += 2; **//Moves the pointer to array[2]**

ptr[1] = 9; **//Sets the value at array[3], which is 17, to 9**

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

cout << &array[5] - ptr << endl; **//Prints out the difference between &array[5] and the location of the pointer, which is &array[2]. Thus, it prints out 3.**

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

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

for (int i = 0; i < 6; i++) **//A for loop used to print out each value of the array**

cout << array[i] << endl;

}

The output for this function is:

3

4

79

-1

9

22

19

5.

void removeS(char\* str);

{

char\* endStr = msg;

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

{

if(\*msg == ‘S’ || \*msg == ‘s’)

continue;

else

{

\*endStr = \*msg;

endStr++;

}

}

\*endStr = 0;

}