Problem # 1

//a

-should be \*(ptr + 1) = 20

- we want to change the value above the starting point to be 20

-there is a space missing in the cout << ‘ ‘ ;

-ptr—should appear at the end of the loop to properly loop through the array and output the values at those locations.

//b

In this problem, the function declaration should be changed to int \* &p for the last parameter in order to function correctly. We needed to pass in the value by reference in order for the function to perform it’s pointer manipulations correctly, and to be able to track the incrementation through the array.

//c

double \* p = new double;

The issue with this problem is that the pointer is not dynamically allocated memory in the main function, and therefore does not work unless this allocation takes place.

//d

If I am understanding the implementation of the while loop in this example, it is attempting to keep looping through as long as it has not reached the ‘\0’, called the null terminator, but does not correctly do this because (str1!= 0 && and str2!=0) does not accomplish this task. 0 is not the null terminator.

A better approach is to use a while loop that compares for char equality(\*str1 == \*str2).

And from there to continue to loop through the elements of the c string and increment the pointers. It is also important to include an if loop to catch the point when both c strings reach the null terminator, otherwise it loops past the end of the strings and causes the function to return that it did not encounter a match.

After breaking from the while loop and and reaching null values, then you know that you had c strings of equal length, and the program can output that it correctly found a match.

//e

In the computeSquares function, the arr allocation should be changed to the following:

int \* arr = new int[10];

The function was not properly allocating memory for the pointer to the arr values it proceeded to create and use.

Problem # 2

//a

string \* fp;

//b

string fish[5];

//c

fp = &fish[4];

//d

\*fp = "yellowtail";

//e

\*(fish + 3) = "mackarel";

//f

fp-=3;

//g

fp[2] = "cod";

//h

fp[0] = "eel";

//i

bool d = (\*fp == fish[0]);

//j

bool b = (\*(fp) == \*(fp +1));

Problem # 3

//a

The way to change the loop to avoid using the increment operator is to switch from a while loop to a for loop, and in the for loop to use the dereferencing operator to add the value that pointer points to plus the ith element forward.

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

{

tot += \*(ptr+i);

}

//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 != chr && \*str != ‘\0’)

{

if ( \*str == chr)

{

return str;

}

}

return nullptr;

}

Problem # 4

#include <iostream>

using namespace std;

// Functions purpose:

// minimart compares two values a and b and returns to the caller the larger of the two values

// returns pointer to int, accepts as arg two pointers to integers(memory locations)

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

{

// compares the values stored in the locations pointed to originaly

// and is comparing to see which is greater

if (\*a < \*b)

return a; // returns adress of a if a is less than b

else

return b; // otherwise returns b

}

// this functions does not work

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

{

// a pointer to type integer is created here, and is assigned the mem. adress of a

int\* temp = a;

// the memory adress of a is assigned the adress of b

a = b;

// b is assigned the adress of a(which was stored in temp)

b = temp;

}

// I believe that this functions fails becuase while it does swap memory adress's

// it does not swap the acutal values

// this function is operational, and proprly performs a swap of two values

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

{

// creates a temprary integer var called temp, and stores in it the value that a points to

int temp = \*a;

// the value at a is assigned the value at b

\*a = \*b;

// the value at b is now assigned the value that was stored in temp, temprarily of course.

\*b = temp;

}

int main()

{

int array[6] = { 5, 3, 4, 17, 22, 19 }; // array of 6 elements created with the following init values

// creates an integer pointer and assigns it the output of the minimart function

// the value outputed by the mini mart function is a adress value

// the function accepts as input the first memory index in array, and the memory value at index 2

int \* ptr = minimart(array, &array[2]); // pointer points to 4 at this point

ptr[1] = 9; // sets the 3rd index location to 9

ptr += 2;

\*ptr = -1; // sets the 4th index location to the value -1

\*(array+1) = 79; // sets the first index location(not the zeroth) to the value 79

cout << "diff=" << &array[5] - ptr << endl; // subtracts the fifth index location from the curent value of ptr

// which here is the 4th index location, hence value of 1

// first swap call does not work

// two calls to swap1 and 2 are performed

swap1(&array[0], &array[1]); // recall that &array[0] same as saying array

// so this functions is the same as the minimart function,

// namely that we are passing in two pointer values

//second swap works

// This is the swap that is responsible for the 4 and the 5 switching positions in the array

swap2(array, &array[2]);

// this function also works the same way, but is operating on different elements in the array

// displays all the values stored in the array by iterating through the index locatins

// via for loop

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

cout << array[i] << endl;

}

Problem # 5

#include <iostream>

using namespace std;

// delete g receives a pointer to the first location in the msg array

void deleteG(char \* src)

{

// while the value pointed to is not the null terminator

while( \*src != '\0')

{

// current char in src compared with desired char's

if(\*src == 'g' || \*src == 'G')

{

// local pointer assigned the location of src

char \* dest = src;

while(\*dest != '\0')

{

//Shift 1 Character to the left, to overwrite

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

//Advance to next position

dest++;

}

}

// loops through the whole src char array that we are pointing to

src++;

}

}

int main()

{

char msg[100] = "I recall the glass gate next to Gus in Lagos, near the gold bridge.";

deleteG(msg);

cout << msg; // prints I recall the lass ate next to us in Laos, near the old bride.

cout << endl;// added this for formatting reasons in my terminal window

return 0;

}