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 + 2) = 10;

**while** (arr+2 >= ptr)

{

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

ptr++;

}

}

b)

Problem: pToMax is not referenced with ptr, so any changes made to pToMax within the findMax function will not change ptr in any way. By adding the ‘&’ operator will fix the problem.

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;

}

}

c)

Problem: ptr does not point to anything. Therefore, \*ncubed

will not work since the ‘\*’ operator has no idea where ptr is pointing to.

int main()

{

int cubeOfFive;

int\* ptr = &cubeOfFive;

computeCube(5, ptr);

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

}

d)

Problem with function: when the variable str1 and str2 are used, they refer to the address of the character that is stored in the memory, instead of the actual character. Therefore, (str1 != str2) will always return true, and str1 == str2 will always return false. The whole function will always return false.

Fix:

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

{

while (\*str1 != 0 && \*str2 != 0) // zero bytes at ends

{

if (\*str1 != \*str2) // compare corresponding characters

return false;

str1++; // advance to the next character

str2++;

}

return \*str1 == \*str2; // both ended at same time?

}

e) getPtrToArray function creates a **local variable**, which is an array of 100 ints. But anArray goes away after we exit the getPtrToArray function, and attempting to access the value pointed to by the ptr variable will result in undefined behavior.

2)

a) double\* cat;

b) double mouse[5];

c) cat = mouse+4;

d) \*cat = 25;

e) \*(mouse+3) = 17;

f) cat -= 3;

g) cat[1] = 42;

h) cat[0] = 54;

i) bool d = (cat == &mouse[0]);

j) bool b = (\*cat == \*(cat+1));

3)

a)

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

{

double tot = 0;

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

tot += \*(scores+i);

}

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)

The output lines are:

**3**

**4**

**79**

**-1**

**9**

**22**

**19**

Because: ptr is initialized to point at the 1st element of the array, since maxwell sets the pointer to point to the address of the element that is larger in maxwell(array, &array[2]).

\*ptr = -1 sets the first element of array to -1. -> {-1, 3, 4, 17, 22, 19}

ptr += 2 makes the pointer point to the 3rd element.

ptr[1] = 9 sets the 4th element of the array to 9. -> {-1, 3, 4, 9, 22, 19}

\*(array+1) = 79; sets the 2nd element of array to 79. ->{-1, 79, 4, 9, 22, 19}

array[5] points to the 6th element, and the 6th element is 3 elements away from the 3rd element, so &array[5] – ptr will be 3.

swap1(&array[0], &array[1]); swaps the address of stored in local variables a & b & temp that points to an integer, but has no effect on the values stored at that index. And such local variables will go away after we exit the swap1 function.

swap2(array, &array[2]); swaps the values stored at array[0] and array[2]-> {4, 79, -1, 9, 22, 19}.

5)

void removeS(char\* msg){

char\* ptr;

while(\*msg != 0){

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

ptr = msg;

while(\*ptr != 0){

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

ptr++;

}

}

else{

msg++;

}

}

}