**Project 6**

**Problem 1**

**A.**

int main() {

int arr[3] = { 5, 10, 15 };

int\* ptr = arr;

\*ptr = 10;

ptr += 1;

\*ptr = 20;

ptr += 1;

\*ptr = 30;

while (ptr >= arr) {

cout << \*ptr << endl;

ptr--;

}

}

**B.**

The error in the findMax function is that the pointer is passed by value instead of being passed by reference. Thus, it creates a copy of the pointer instead of the actual pointer itself. To fix this, simply change the pointer to be passed by reference.

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

{

if (n <= 0)

return;

pToMax = arr;

for (int i = 1; i < n; i++)

{

if (arr[i] > \*pToMax)

pToMax = arr + i;

}

}

**C.**

The error in the main function is that the pointer variable is declared but not initialized, so does not point at anything in memory. Simply initializing a variable to hold the int you want cubed solves the problem. We then proceed to initialize the pointer to the address of that variable, and subsequently pass the int variable to the computeCube.

int main() {

int n;

int\* ptr = &n;

computeCube(5, ptr);

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

}

**D.**

The problem with strequal is the way the array comparison is taking place. The function is not comparing each individual character, instead the base addresses of the array in memory, so it will always evaluate to false. This can be solved by passing pointers to the arrays. We can then proceed to dereference the pointers to check for the zero byte, and dereference the pointers to compare the values. Ultimately, we dereference the pointers for the final comparison.

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

{

while (\*str1 != 0 && \*str2 != 0) {

if (\*str1 != \*str2)

return false;

str1++;

str2++;

}

return \*str1 == \*str2;

}

**E.**

The problem with this program is the scope of the anArray variable in the getPtrToArray function. AnArray is a local variable, so the pointer points to the first element of that array, but upon the completion of the function, all elements within anArray are discarded and set to undefined values. The program fails because the behavior is undefined, resulting in ptr[i] outputting random values.

**Problem 2**

**A.**

double\* cat;

**B.**

double mouse[5];

**C.**

cat = mouse + 4;

**D.**

\*cat = 25;

**E.**

\*(mouse + 3) = 54;

**F.**

cat -= 3;

**G.**

cat[1] = 17;

**H.**

cat[0] = 42;

**I.**

bool d = cat == mouse;

**J.**

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

**Problem 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) {

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

if (\*str == chr)

return str;

}

return nullptr;

}

**Problem 4**

Program Output

3

4

79

-1

9

22

19

Explanation

// The array is initialized to be { 5, 3, 4, 17, 22, 19 }.

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

// A pointer maxwell ptr is initialized to be maxwell(array, &array[2]), which points to array[0], since maxwell() takes in two pointers and returns the one that points to a bigger value.

int\* ptr = maxwell(array, &array[2]);

// \*ptr, or array[0] is set to -1. The array is now { -1, 3, 4, 17, 22, 19 }.  
\*ptr = -1;

// ptr now points to array[2].  
ptr += 2;

// ptr[1], which is array[2+1], is set to 9. The array is now { -1, 3, 4, 9, 22, 19 }.  
ptr[1] = 9;

// \*(array+1), which is array[1], is set to 79. The array is now { -1, 79, 4, 9, 22, 19 }.  
\*(array+1) = 79;

// Since ptr == array + 2, and &array[5] == array + 5, this prints 3.

cout << &array[5] - ptr << endl;

// Here, swap1 does not actually cause a change. It only exchanges variables local to its function and never dereferences the pointers given to it.

swap1(&array[0], &array[1]);

// swap2 swaps the elements at indices 0 and 2, or array[0] and array[2]. The array is now { 4, 79, -1, 9, 22, 19 }.

swap2(array, &array[2]);

// Finally, each element of the array is printed on a newline.

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

cout << array[i] << endl;

**Problem 5**

void removeS(char\* ptr) {

char\* result = ptr;

for (; \*ptr != 0; ptr++) {

if (\*ptr != 'S' && \*ptr != 's')

{

\*result = \*ptr;

result++;

}

}

\*result = 0;

}