**1.**

**a.**

The statement \*ptr + 1 = 20; does not work since it should increment the pointer first before dereferencing it. The original while loop prints the values in the opposite order.

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 -= 2;

while (ptr < arr + 3)

{

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

ptr++;

}

}

**b.**

The pointer pToMax is not passed to the function findMax by reference, so a new local pointer variable is created. We need to pass by reference so ptr in the main function is updated.

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;

}

**c.**

The computeCube function does not work because the pointer ptr is not pointing to anything when it is initialized so we are not allowed to set its value.

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int cubed = 0;

int\* ptr = &cubed;

computeCube(5, ptr);

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

}

**d.**

When comparing characters, the function should use the dereference operator, instead of the pointer to the array. Otherwise, only the addresses would be compared.

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?

}

int main()

{

char a[15] = "Noor";

char b[15] = "Noah";

if (strequal(a,b))

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

}

**e.**

The program is declaring anArray as a local variable in the function getPtrToArray and then returning the pointer to the local array as its return value. After the function has finished execution, anArray is not considered important by the program anymore so it could be overwritten with other values such as while executing function f. This means that the original values in the array could be overwritten.

**2.**

1. double\* cat;
2. double mouse[5];
3. cat = mouse + 4;
4. \*cat = 42;
5. \*(mouse + 3) = 25;
6. cat -= 3;
7. cat[1] = 17;
8. cat[0] = 54;
9. bool b = (\*cat==\*(cat+1));
10. bool d = (cat==&mouse[0]);

**3.**

**a.**

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

{

const double\* ptr = scores;

int index = 0;

double tot = 0;

while (index < numScores)

{

tot += \*(ptr + index);

index++;

}

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.**

#include <iostream>

using namespace std;

int\* maxwell(int\* a, int\* b) // returns the pointer that holds the greater int

{

if (\*a > \*b)

return a;

else

return b;

}

void swap1(int\* a, int\* b) // swaps the address of pointer a with b but does nothing because

{ // they aren’t passed by reference

int\* temp = a;

a = b;

b = temp;

}

void swap2(int\* a, int\* b) // swaps the value at pointer a with that at pointer b

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main()

{

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

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

\*ptr = -1; // the value at ptr or array[0] is -1

ptr += 2; // ptr is equal to &array[2]

ptr[1] = 9; // the value at array[3] is now 9

\*(array+1) = 79; // array[1] is 79

cout << &array[5] - ptr << endl; // pointer subtraction 5-2 = 3 which is printed on a line

swap1(&array[0], &array[1]); // does nothing since the pointers aren't passed by reference

swap2(array, &array[2]); // swaps the values at array[0] and array[2]

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

cout << array[i] << endl; // each value of the array is printed on a line

}

After each statement, the value of the array is

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

{ 5, 3, 4, 17, 22, 19 }

\*ptr = -1;

{ -1, 3, 4, 17, 22, 19 }

ptr += 2;

{ -1, 3, 4, 17, 22, 19 }

ptr[1] = 9;

{ -1, 3, 4, 9, 22, 19 }

\*(array+1) = 79;

{ -1, 79, 4, 9, 22, 19 }

cout << &array[5] - ptr << endl; (5-2=3, so 3 is printed out)

{ -1, 79, 4, 9, 22, 19 }

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

{ -1, 79, 4, 9, 22, 19 }

swap2(array, &array[2]);

{ 4, 79, -1, 9, 22, 19 }

The for loop prints out the values of the array so the final output would be

3

4

79

-1

9

22

19

**5.**

void removeS(char str[])

{

char\* ptr = str;

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

if (tolower(\*str) != 's')

{

\*ptr += \*str;

ptr++;

}

}