1a.

**int** main()

{

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

**int**\* ptr = arr;

\*ptr = 30; // set arr[0] to 30

ptr++;

\*ptr = 20; // set arr[1] to 20

ptr++;

\*ptr = 10; // set arr[2] to 10

**for** (ptr = arr; ptr < arr+3 ; ptr++)

{

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

}

}

1b.The function doesn’t work because the pointer is originally passed by value. In order to fix the function, I added an ampersand in front of the pointer variable pToMax so that it would be passed by reference, and therefore, the new pointer value can be usedin the main function.

**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;

}

1c. Originally, the pointer is never pointing to any address. To fix this, I initialized and declared an int variable and assigned the pointer to point to it.

**void** computeCube(**int** n, **int**\* ncubed)

{

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

}

**int** main()

{

**int** hello = 5;

**int**\* ptr = &hello;

computeCube(5, ptr);

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

}

1d. The function doesn’t work because all its comparisons are between two addresses instead of two values. The compared characters may be the same but their addresses will be different. To fix this, the comparisons must be made between the values of where the pointer is.

// return true if two C strings are equal

**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] = "Chen";

**char** b[15] = "Cheng";

**if** (strequal(a,b))

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

}

1e. The getPtrToArray function creates an array that can’t be accessed outside of that function. Therefore, when the main function ouputs ptr[i], it can’t access the array from getPtrToArray, and instead, outputs a garbage value.

2.

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

3a.

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

{

**double** tot = 0;

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

{

tot = \*(scores + i);

}

**return** tot/numScores;

}

3b.

**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**;

}

3c.

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

{

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

{

**if** (\*str == chr)

**return** str;

str++;

}

**return** **nullptr**;

}

4. The program outputs:

**3**

**4**

**79**

**-1**

**9**

**22**

**19**

**int** main()

{

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

// set ptr to point to array[0] since 5 > 4

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

// array[0] == -1

\*ptr = -1;

// ptr points to array[2]

ptr += 2;

// array[3] == 9

ptr[1] = 9;

// array[1] == 79

\*(array+1) = 79;

// output address of 6th element minus address of ptr (5-2)

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

// does nothing bc pointers passed by value, not reference

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

// array[0] == 4, array[2] == -1

swap2(array, &array[2]);

// print values in element on separate lines

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

cout << array[i] << endl;

}

5.

**void** removeS(**char**\* cstring)

{

**if** (\*cstring == '\0')

**return**;

**while** (\*(cstring + 1) != '\0')

{

**if** (\*cstring == 's' || \*cstring == 'S')

{

**for** (**char**\* ptr = cstring; \*ptr != '\0'; ptr++)

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

}

**else**

{

cstring++;

}

}

**if** (\*cstring == 's' || \*cstring == 'S')

\*cstring = '\0';

}