**1A:**

int main()

{

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

int\* ptr = arr;

\*ptr = 10;          // set arr[0] to 10 instead of 30

\*++ptr = 20;   // set arr[1] to 20, use increment operator

ptr += 2;

ptr[-1] = 30;     // set arr[2] to 30 instead of 10, change subscript to -1

while (ptr > arr)   // set to > instead of >=

{

    ptr--;

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

}

}

**1B:**

The problem is that the function only modifies the local parameter that pToMax points to. In order to properly fix this, we must include the ampersand because it modifies a reference to the pointer that was passed as a parameter to findMax. Now, when the array is modified, so is ptr, which is in the main function.

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

{

if (n <= 0)

    return;  // no items, no maximum!

pToMax = &arr[0];

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

{

    if (arr[i] > \*pToMax)

        pToMax = &\*(arr + i);

}

}

**1C:**

The problem is that ptr does not point to any specific value. To fix this I made ptr point to the location of an intx. I did this by adding an & preceding x, which allowed me to set ptr equal to &x. Since ptr now points to x, when function computeCube() is called, x will be set to 125.

int main()

{

int x=0;

int \* ptr=&x;

computeCube(5, ptr);

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

}

**1D:**

The first major issue was, when using str1 and str2 as parameters, the derefercing operator was not used (add \* preceding str1 and str2 while used as conditions). Without the dereference operator, the function will compare the entire array instead of each character of the C strings.

// return true if two C strings are equal

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

{

while (\*str1 != 0 && \*str2 != 0) // checks for zero byte

{

    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] = "Zhou";

char b[15] = "Zhu";

if (strequal(a, b))

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

}

**1E:**

The issue with the code is that the call to getPtrToArray in the main function will return random values as the array. This is due to the fact that the array is created in the getPtrToArray function; all memory that is allocated to storing the values of the array is cleared before the array is outputted.

**2:**

double \*cat; //a

double mouse[5]; //b

cat = &mouse[4]; //c

\*cat = 25; //d

mouse + 3 = 42; //e

cat -= 3; //f

cat[1] = 54; //g

cat[0] = 27; //h

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

bool d = cat == &mouse; //j

**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)

{

for (const char \*cur\_char = str; \*cur\_char != 0; cur\_char++)

    if (\*cur\_char == chr) return cur\_char;

return nullptr;

}

**4:**

a) Set ptr equal to address of paramaeter given

b) Compares which value has greater value. Array[0]>array[2] (5>4)

c) Array[0] is set to -1

d) Ptr incremented by 2

e) The element proceeding what ptr points to is set to 9, array[3]=9

f) 1st element of array set to 79

g) Print the difference between memory address of 5th index and current element (4). Therefor 5-2=3

h) Swap1 makes a swap in memory locations 0 and 1. Since there is no reference, the swap is done locally. No change in the array.

i) Swap2 makes swap in values stored at respective memory locations. Index 0 and 2 swapped.

j) Resulting array{ 4, 79, -1, 9, 22, 19 }

**5:**

void removeS(char \*parameter)

{

   do

   {

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

       {

           char \*i;

           for (i = parameter; \*i != 0; i++)

           {

               \*i = \*(i + 1);

           }

           \*(i - 1) = 0;

       }

       else

       {

           parameter++;

       }

   } while (\*parameter != 0);

}