Lab 12 - Arrays and Pointers

1. Creating a Sequence

Create a program named `arrayLab.cpp`

1. Write a function named `displayArray()` which accepts two parameters: an array of integers, and the size (number of elements) of the array. The function prints the elements of the array to one line on the screen.

   - Sample `main()` to call it:
     ```c++
     int main() {
     // Test displayArray():
     int arr0[] = {42, 0, 1, 101, 58};
     displayArray(arr0, 5);
     return 0;
     }
     ```
   - Sample output:
     ```
     Array contents: 42, 0, 1, 101, 58,
     ```

2. Write a function of the following prototype:

   ```
   void populateSequence(int arr[], int size, int start, int gap);
   ```

   The function fills `arr[]` with a sequence of integers starting at `start`, and incrementing by `gap`. The function must `return nothing` and `output nothing`: it populates the `arr[]` array, which changes the actual array it is passed by the calling code.

   - **Hint:** Use a `for` loop to count how many values you have put into the array. 
   (A `for-each` loop does not work for an array in this situation).

   - The following is an example of how to use the function. You should test with at least four more examples to satisfy yourself that your implementation is correct.
     ```c++
     int main() {
     // Test 1 populateSequence: Expected values 7, 9, 11, and 13
     int arr1[4];
     populateSequence(arr1, 4, 7, 2);
     displayArray(arr1, 4);

     // Test 2 populateSequence: Expected values 2, -1, and -4
     int arr2[3];
     populateSequence(arr2, 3, 2, -3);
     displayArray(arr2, 3);
     return 0;
     }
     ```

3. Understanding:

   - Most functions use a return value to pass back information. However, 
   `populateSequence()` has a `void` return type, yet it is able to give the calling code 
   (in this case `main()`) some values in the array. Explain how this happens. Why can 
   this function use the `void` return type?
2. Find Smallest Value

Still in arrayLab.cpp:

1. Write a function named `minValue()` which:
   - accepts 2 parameters: an array of int's, and the number of elements in the array.
   - returns the smallest value that is in the array
   
   The following code uses the `minValue()` function:
   ```
   // Test for minValue
   int arr3[] = {17, 3, 12, 11, 4};
   int val1 = minValue(arr3, 5);
   cout << "Min1: " << val1 << "\n";
   // should print the value 3
   
   int arr4[] = {1, 0, 7, 23, 2, -1};
   cout << "Min2: " << minValue(arr4, 6) << "\n";
   // should print the value -1
   ```

   Implementation Hints:
   - Create a variable to hold the smallest value that you have found so far.
   - Use a for loop to cycle through each value in the array. If it's smaller than the smallest you've found so far, then you have a new smallest value.
   - What should your variable for “smallest-value-so-far” be initialized to? What is the problem with starting with 0? With -1? With -100000?
   - Pseudocode for the algorithm:
     ```
     smallest = arr[0]
     for every value in the array
       if current value < smallest
         smallest = current value
     return smallest
     ```

2. Understanding:

   - For the `minValue()` function, why start the “smallest” variable at `arr[0]` instead of the number 0?
   - What happens in your program when there are multiple values which have the same minimum value? Does this cause a problem? Why or why not?
   - Explain why the `minValue()` function has a return type instead of using the same method to pass back information as `populateSequence()`.
   - Explain what the output (to the screen) of the following code is:
     ```
     int arr[] = {10, 20, 0};
     minValue(arr, 3);
     ```
     - Explain what changes are needed to display the answer it computes to the screen.
     - Why is it better to have the function return a value, rather than directly print the value to the screen?
3. Pointers

Make a new file named `pointers.cpp`.

1. Create a function which checks if there are any negative numbers in an array of doubles. Accepts 2 parameters: a pointer to an array of doubles, and the number of elements in the array. Return `true` if there are any negative numbers in the array, `false` otherwise.
   - It accepts a pointer, not an array, so the prototype must be:
     ```
     bool hasNegative(double *arr, int size);
     ```
   - *Hint: Even though you are taking in a pointer, you can still use array syntax on it: [*]

2. Test your function using these as a starting point:
   ```
   double arr1[] = {2, 5.2, 6, 8, 6, 10, 325532, 0};
   cout << "Has negative #1?: " << hasNegative(arr1, 8) << endl;

   double arr2[] = {2, 7.2, 0.1, -2, 5};
   cout << "Has negative #2?: " << hasNegative(arr2, 5) << endl;

   double arr3[] = {-1, -5, -153};
   cout << "Has negative #3?: " << hasNegative(arr3, 3) << endl;
   ```

3. Create a function named `zeroDissimilarPrefix()` which accepts two pointers to integers (not arrays), and returns nothing. (It also generates no output)
   - The function compares digits of each of the values pointed to by the arguments. It starts with the least significant digit (the right-most digit in each value). When the digits differ from each other, it zeros out that digit and all digits to the left, and returns.
   - It analyzes and changes at most 10 digits.
   - For example, when passed pointers which point to the values 12345 and 412945, it changes each to be 45. This is because their third least significant digits differ, and hence the third digit and all digits to the left are zeroed out.
   - Other examples (two values passed in on left, what they are changed to on right):
     - 1 and 100: set both to 0
     - 0 and 101: set both to 0
     - 1 and 101: set both to 1
     - 24680 and 68: set both to 0
     - 24680 and 80: set both to 80
     - 24680 and 8080: set both to 80
   - *Algorithm Hint: You are searching for the smallest power of 10 such that each parameter mod this power of ten gives a different answer.*
Implementation Hint:

Start with 1 as your current power of ten. Check if both parameters, mod this power of ten, are the same.

- If they differ, you are done searching: use mod to retain only the portion of the number to the right of this digit.
- If the mod’ed values are the same, multiply the current power of ten by 10, and repeat the above check.

Check up to 10 digits
Mod example to help: 12345 mod 10 (10^1) gives 5, mod 100 (10^2) gives 45; mod 1000 (10^3) gives 345...

You may find the following function useful for testing:

```
const int NUM_DIGITS = 10;
void testZeroPrefix(int a, int b)
{
    cout << "Test on \" << setw(NUM_DIGITS) << a
    " and \" " << setw(NUM_DIGITS) << b << endl;
    zeroDissimilarPrefix(&a, &b);
    cout << " = \" << setw(NUM_DIGITS) << a
    " and \" " << setw(NUM_DIGITS) << b << endl;
}
```

Hint: Have `main()` call `testZeroPrefix()` with different parameters, such as:

```
testZeroPrefix(123456789, 122456789);
```

4. Sample output (calling `testZeroPrefix()` multiple times, each with different arguments).

Has negative #1?: 0
Has negative #2?: 1
Has negative #3?: 1
Test on 1 and 1 = 1 and 1
Test on 4 and 5 = 0 and 0
Test on 56 and 56 = 56 and 56
Test on 11 and 12 = 0 and 0
Test on 25 and 45 = 5 and 5
Test on 234 and 299 = 0 and 0
Test on 789 and 299 = 9 and 9
Test on 699 and 299 = 99 and 99
Test on 123456789 and 122456789 = 456789 and 456789
5. **Understanding:**
   - What two ways can an array be passed to a function?
   - How a function can accept pointers, access and change the values stored to the location they point to.
   - How to pass a function the address of a local variable (or a parameter).