Homework 10 – Arrays

For this homework, you will read up to 1000 long integer values from a file into an array, find the max, min and the i^{th} value. Then you will sort the array (examples will be shown in class), and finally print the max, min and i^{th} value for the sorted array. Some input files will contain more than 10000 numbers; you need to stop reading at 1000!!!

Getting Started:

To get started, create a new project and copy-paste the code from the bottom of this assignment to the main.cpp file in the new project. Note that the code must be modified significantly for the new program. There are lots of places where you have to add **lines** of code and other places where you have to insert pieces of code into lines (like in if conditions).

The code must prompt the user for an input filename and a value that is used as the position of a number in the **array** to display. For example, if the user enters 100, then you will print the 100^{th} number in the **array** as part of the output. If the user enters a number greater that the number of values in the array, then you print an error message (see below).

Requirements:

This is extremely important: I specify in which functions certain code must be implemented; this is so that you learn how to use functions, arguments, parameters, arrays, etc. For example, you must declare the array of 1000 integers in main() and pass it as a parameter to both the readArray() and sortArray() functions, see below for the prototypes and descriptions for the functions. You must use (and write) these functions; you may not use the Standard Template Library (STL) to sort the number. Because arrays are passed "like" by-reference, you use the array declared within main() throughout the program; you share the memory space of the array declared in main() as though it were a reference parameter. You should declare a constant for the size of the array, 1000, and use it everywhere in the code where you would use 1000.

`int main()`

1. **Declare** the required variables (e.g. the array) and other variables you need (like min, max, etc.).
2. **Prompt** for file name and open the file. If it's not able to, print out an error message.
3. **Prompt** for the value to display, and if necessary **display "xxx is bigger than 1000!"** where **xxx** is the number the user entered. See the example below.
4. **Call** readArray() with appropriate parameters. Assign the return value of readArray() to a variable so you can have if statements that output any needed errors or warnings, which are described below in green.
5. **Print** the First, Last, and i^{th} values in the unsorted array.
6. **Call** sortArray()with appropriate parameters.
7. **Print** the Min, Max and i^{th} values in the array. Remember the values are sorted in the array.
8. You **must print your output from the main() function**. You **may not** use `couts` in the `readArray()` and `sortArray()` functions.

`int readArray(ifstream& ifile, long arr[]);`

- **Read** up to 1000 longs into the array that is passed as a parameter. The file may contain more than 1000 numbers; it may contain fewer. Your program may crash if you read more than 1000 numbers.
- The values in the input file will be whitespace-separated whole numbers that fit into a long. Use extraction operator(s).
- **Return** the number of values successfully put into the array. A return value of 1000 means the function call was successful. If any other number is returned, then main() should print a warning message, `"WARNING: Only 123 numbers were read into the array!"` where 123 is the number successfully read. If the value requested by the user is greater than the number read in, then main() should print an error message that says: `"There aren't that many numbers in the array!"`

```
void sortArray(long arr[], int numberInTheArray);
```

- **Sort** the array arr in ascending order, lowest to highest. **You should not declare another array inside this function**. Returning an array that is declared within the sortArray() function is a logic error that can make your program work sometimes but not work other times. Logic errors compile and run, but they produce incorrect results. You may not use a library sort routine; you must code it yourself in this function.
- The second argument is the number of elements to sort, since there may not be 1000 elements within the array. If you sort more than that number of elements, then your sorted numbers will include extra "garbage" numbers, which gives you incorrect results.
- After this function call, the first number in the array is the smallest number in the array; the last is the largest. you do not need to "find" the max and min.

A sample input will be provided, but you should make your own test files so that you know the output when you run your program to output different i^{th} values. You need to also test for files with fewer values in them, like 1 or 2.

How you should incrementally develop the code:

You can actually develop the code any way you want to as long as it meets the specifications, but **the TAs can most easily help you if you follow the development plan outlined here.** The TA's may ask you to show them the outputs for the previous steps when you ask for a later step. They can ask you to implement those previous steps before they can help you. If you come for help on #15 but not have implemented the #1-14, they cannot help you with the error message.

1. **Use the code outline given below.**
2. **To begin writing the code, assume for now** all data is valid and there are at least 1000 numbers in the input file. After you get the program working under the assumption of 1000 inputs number, then you must go back and modify the code to work for fewer input numbers.
3. Declare the array and other variables in main(). **// given below**
4. Read the filename and number to display (i^{th} value) as described previously. **// given below**
5. If you are using the given code, go thru and put in dummy value to make the code compile, like `true` in if statement conditions or 3 where I have given you incomplete output values.

6. Compile and run your program. You might read 1 value from the file to **make sure you have opened the file and you can read a single value from it. Then, delete the single read.**
7. Add the readArray() function to read in the data. Again, assume there are at least 1000 numbers in the file for now.
8. Print the array inside the function **as you are reading in the numbers inside the loop to check if the code is working up to this point.**
9. Remove the cout from #8 inside the loop, and put a simple `for` loop at the end of the readArray() function to print the values in the array.
10. Move the `for` loop from #9 and put it into main() **to make sure your code is able to print out the array inside of main(). Then,** once you confirm that you can print the array, comment out this print loop.
11. Add the code inside main() to find and print the first, last, and i^{th} value of the array.
12. Print those values and test your code with different values for **`i`** (like 1, 1000, and 3) to **ensure that you are properly finding the ith value.**
13. Add the sortArray() function to put the number in the array in order.
14. Using another simple for loop, print the array inside the sort function to **make sure that it is properly sorted.**
15. Comment out the cout lines from #14 and copy it into main() to **make sure the sorted array is being passed back properly.**
16. Print the min, max, and i^{th} values. (Min is the first number in the array after the sort; max is the last!) You must not have code to look for max and min as you are reading in the number; you don't need to!
17. Now, modify the code inside readArray() to check that you haven't reached the end of the file as well as reading up to 1000 numbers. I will talk about this in class. For this, you then return the number of elements you successfully read into the array.
18. Add the `if` statements in main() to check the return value and print out the error and warning messages.
19. The return value of readArray() is the number of values put into the array, which can be less than 1000 if the file is smaller. Pass this return value of readArray() as a parameter to sortArray() so that you are only sorting the first x number of values, where x is the return value of the read.
20. Modify sortArray() so that is only sorts the first numberInTheArray elements.
21. **Modify printing the min, max, and i^{th} values for this change also.** The max value is no longer the last value in the array if you don't have 1000 items in the array!
22. Make sure to thoroughly test your code before you start submitting to Web-CAT!
23. Please remember to follow coding and commenting guidelines.

Sample Output

Here are 2 sample outputs so that you know what to cout. User input is in **blue**.

Sample 1 (There are 2 other numbers smaller than 44. 1000 numbers were read into the array.):

```
Input File Name: input.txt
Which number do you want to return? 3

Before Sort:
 First is {1101}.
 Last is {11}.
 Value 3 is {225}.

After Sort:
 Min is {-111}.
 Max is {544245}.
 Value 3 is {44}.
```

Sample 2 (We are only handling 1000 numbers.):

```
Input File Name: anotherInput.dat
Which number do you want to return? 1111

Before Sort:
 First is {1910}.
 Last is {1}.
 1111 is bigger than 1000!

After Sort:
 Min is {1}.
 Max is {544245}.
 1111 is bigger than 1000!
```

Sample 3 (There are 453 numbers in the input file.):

```
Input File Name: anotherInput2.dat
Which number do you want to return? 900

Before Sort:
 First is {1}.
 Last is {544245}.
 WARNING: Only 453 numbers were read into the array!
 There aren't that many numbers in the array!

After Sort:
 Min is {1}.
 Max is {544245}.
 WARNING: Only 453 numbers were read into the array!
 There aren't that many numbers in the array!
```

Sample 4 (There are 453 numbers in the input file, and the user asked for a number less than that.):

```
Input File Name: anotherInput2.dat
Which number do you want to return? 90

Before Sort:
 First is {-9991}.
 Last is {544245}.
 WARNING: Only 453 numbers were read into the array!
 Value 90 is {19}.

After Sort:
 Min is {-9991}.
 Max is {544245}.
 WARNING: Only 453 numbers were read into the array!
 Value 90 is {144}.
```

Sample 5 (empty file):

```
Input File Name: empty.dat
Which number do you want to return? 111

The file is empty!
```

Sample 6 (no file):

```
Input File Name: Input.dat

That file does not exist!
```

Code Outline (starter)

```
#include <iostream>
#include <fstream>

using namespace std;

// prototypes & constants
int readArray(ifstream& ifile, long arr[]);
void sortArray(long arr[], int numberInTheArray);

const int LG = 1000;  // You can change the value of LG while you are
                      //  developing your program to test smaller data files.

int main()
{
      //Declare variables
      string infile2; //user's inputed file name
      ifstream myInfile;
      long inputArray[LG];
      long ith; //user's inputed value to print ith value
      int number = 0; //value that gets returned by readArray()

      //Prompts user for file
      cout << "Input File Name: ";
      cin >> infile2;

      //Open the input file
      myInfile.open(infile2.c_str());

      //Check to see if infile is okay
      if ( !        )
      {


      }

      //Ask user which number they want to return
      cout << "Which number do you want to return? ";
      cin >> ith;

      //call readArray
      number =

      // Output after readArray but before sortArray
           // Print first, last, etc.
```

```
if ( ?????  )   // if there are less than LG numbers in the file
{
    cout << "\nBefore Sort:" << endl;
         // print first & last
         // print warning message
         // print "Value XXX is" OR error message
    if (   ???   )    // if number read is less than or = the ith value,
                      //      then you can print Value ith is { }
    {

    }
    else // print 'not that many numbers'
    {

    }

    //call sortArray



    //Outputs after sortArray
    // Print min/first, max/last, and amount of numbers in the array AFTER the sort
    cout << "\nAfter Sort:" << endl;
         // print first & last
         // print warning message
         // print "Value XXX is" OR error message
    if (   ith <= number   ) // same if/else as above to print Value
    {

    }
    else
    {

    }
}

//   --the array is full
// if the amount of numbers in the file is larger than LG (the const array size)
// data printed shows the min, max and value, but lets the user know the value is
// larger than the constant value
else
{
    // Prints first & last
    cout << "\nBefore Sort:" << endl;
         // print first & last

    if (   ???   )   // print "Value XXX is" OR error message
    {

    }
    else
    {

    }

    //call sortArray
```

```
         //Outputs after sortArray
         cout << "\nAfter Sort:" << endl;

              // print min (which is first) and max wich is last

         if (   ???   )  // print "Value XXX is" OR error message
         {

         }
         else //(ith > LG)
         {

         }
     }

     //Close the input file
     myInfile.close();

     return 0;
}  // en of main()


/** Reads the array from infile
 *
 * @param function takes the user choice as an ifstream (ifile)
 *@param function takes the user choice as a long (arr[])
 *
 */
int readArray(ifstream& ifile, long arr[])
{
     int count = 0;
   //  count  < LG && ifile



}

/** Bubble sort the array
 *
 * @param function takes the user choice as a long (arr[])
 * @param function takes the user choice as an int(numberInTheArray)
 *
 */
void sortArray(long arr[], int numberInTheArray)
{
     //initialize variables
     int temp = 0;

     //for loop sorts array from 0 to numberInTheArray-1
     for (      )
     {
         for (   )
         {

         }
     }
```

```
}
```

Submit your Solution to Web-CAT