Part 1, Write And Test An Array Class [MyStaticArray.h]

Write a data structures template. The resulting template can be used in any program in place of a C++ array, without having to copy/paste the class code the way you did with class Array in Assignment 1.

**Requirements.** Develop **MyStaticArray.h** as you write a test driver CPP with **class MyStaticArray**, defined and *fully tested*. Write the public interface exactly as specified below -- do not add to, or change the public interface as specified.

1. Write the template for an array of values of unspecified type.
2. Let the capacity be set as part of the template specification, as modeled in the reading.
3. Include a square bracket getter and a setter, both with index range-checking, returning whatever value you wish, if out of range.
4. Include a getter named **MyStaticArray::capacity( )** to return the data structure's capacity.
5. Initialize *all* data members to their default values in the constructor. That includes setting the array elements to their default data type.
6. Do tests with int, double, or char. Also do tests with an object, like string.

Do *NOT* write any other functions in the public interface.

Part 2, Write An Array Application

Write a console app using your **MyStaticArray template**. Use your already-tested and verified H file from part 1.

Exactly as in Lab Assignment 1, *this* app lets its user enter as many values as they like, and when that process is completed, lets the user look up values by matching index. Except that these values are *doubles*. Use a **MyStaticArray** object of capacity 100 to track the values.

In a loop, the app should prompting the user to enter a *pair of numbers* on the same line: a whole number *index* and its corresponding ***floating point*** *value*. Do *not*validate input in the app, because your template should handle out-of-range indexes, and it should allow overwriting an already-entered index. Quit the loop when an *uppercase* or *lowercase* Q is entered for *either* the index or the value. Indexes can be entered in any order -- they don't have to start with zero and go up by one thereafter. It's whatever the user enters.

Your app should keep track of which indexes got entered. Use a *bool***MyStaticArray** for that.

After all data entry is complete, the app should:

1. output how many (unique) indexes got entered,
2. output the list of all used indexes and their values, per the example below, and
3. implement an event-controlled loop that prompts for an index value and outputs whether the index is in use or not, and if in use, what is the value stored for that index. Loop until the user elects to stop by entering uppercase or lowercase Q.

Here's a sample of how this should work (user input in blue):

Input an index and a value [Q to quit]: 33 1.2 Input an index and a value [Q to quit]: 4 0 Input an index and a value [Q to quit]: 5 300 Input an index and a value [Q to quit]: x 1.7 Input an index and a value [Q to quit]: 33 120 Input an index and a value [Q to quit]: -1 23.4  
Input an index and a value [Q to quit]: 2000 -999 Input an index and a value [Q to quit]: q You stored this many values: 4 The index-value pairs are: 0 => 1.7 4 => 0 5 => 300 33 => 120 Input an index for me to look up [Q to quit]: 33 Found it -- the value stored at 33 is 120  
Input an index for me to look up [Q to quit]: 38  
I didn't find it  
Input an index for me to look up [Q to quit]: 0  
Found it -- the value stored at 0 is 1.7  
Input an index for me to look up [Q to quit]: -100  
I didn't find it Input an index for me to look up [Q to quit]: 1000 I didn't find it Input an index for me to look up [Q to quit]: Q

Design the prompts and the output formatting as you like.

Submit 4 Files

Submit the 2 CPPs (the test driver and the app(main.cpp)) and the H file for grading.

Also, include a word doc file with your code passted on it and a screenshot of the application otput.