Data Structures & Algorithms (Course & Lab) – Fall 2015

(BS-SE-F14 Morning & Afternoon)

PRACTICE Assignment # 3

Submission Deadline: None!!

Instructions

- **Although this is just a practice assignment, but still you need to work on it individually. Absolutely NO collaboration or discussion is allowed.**
- **You are not required to submit this assignment, but I will assume in quizzes and exams that you have completed this assignment.**

Following problem description is taken from the Pages 302 and 303 of the book "Data Structure and Algorithms in C++" (4th Edition) by Adam Drozdek:

A binary tree can be used to sort *n* elements of an integer array. First, create a full binary tree, a tree with all leaves at one level, whose height $h = \lceil \lg n \rceil + 1$, and store all elements of the array in the first *n* leaves. In each empty leaf, store an element *E* greater than any element in the array.

Figure 1 shows an example where the array to be sorted is: {8, 20, 41, 7, 2}. Height of the tree is $h = \lceil \lg 5 \rceil + 1 = 3 + 1 = 4$, and *E* = 42.

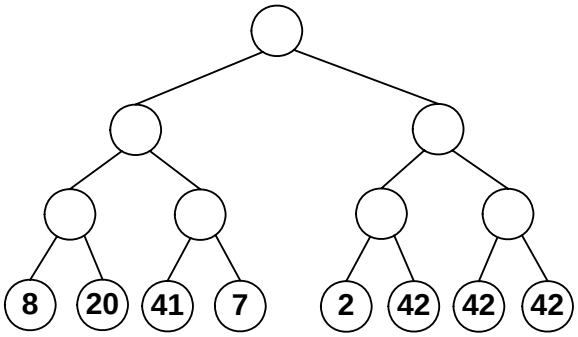

Figure 1

Then, starting from the bottom of the tree, assign to each non-leaf node the minimum of its two children values, as shown in Figure 2, so that the smallest element *min* in the tree reaches the root of the tree.

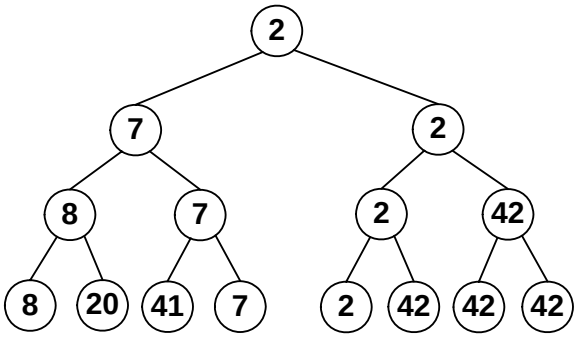

Figure 2

Next, until the element *E* is assigned to the root, execute a loop that in each iteration stores *E* in the leaf which contained the value min, and that, also starting from the bottom, assigns to each node the minimum of its two children. Figure 3 displays this tree after one iteration of the loop (note that the second smallest element has now been moved to the root).

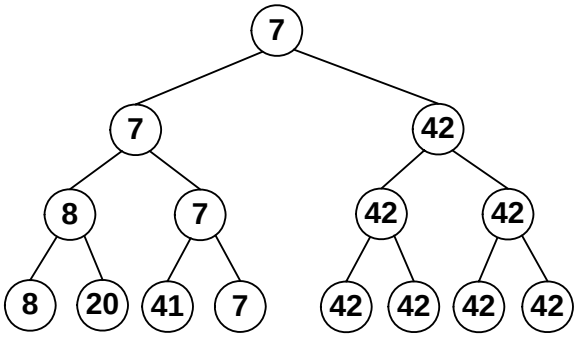

Figure 3

In this assignment, you are required to implement a sorting algorithm using the technique mentioned above. Since, the tree used in the above-mentioned sorting algorithm is a full binary tree, you should use an **array-based implementation of binary trees**.

Design and implement a class **`FullBinaryTree`** which will use an array to store the elements of the tree. Data members of this class will be:

```
class FullBinaryTree {
  private:
    int * tree;     // Array containing the elements of full binary tree
    int treeSize;   // Size of the array 'tree'
    int * sorted;   // Array to store the numbers in sorted order
    int sortSize;   // Size of the array 'sorted'
    int E;          // A value larger than all the elements which are to be sorted
};
```

The class **`FullBinaryTree`** should provide the following functions:

`Constructor`	To create a full binary tree of an appropriate height. Constructor will take the total number of elements (*n*) as an argument, and it will allocate the arrays **`tree`** and **`sorted`**, and initialize **`treeSize`** and **`sortSize`** accordingly.
`Destructor`	To deallocate the tree (array).
`loadValues`	This function will take an integer array and its size as arguments and loads the values of that array into the first *n* leaves of the full binary tree. The remaining leaves will be loaded with a value **`E`** which is greater than any element in the array. (*Note: This function will need to determine the value of* **`E`** *and initialize it properly*).
`sortValues`	This function will sort and display the values present in the leaves into *ascending* order using the afore-mentioned algorithm. The values in sorted order will be put into the array **`sorted`**. During the process of sorting this function should also display the contents of the array **`tree`** after each iteration of the loop (after the next smallest element has been moved into the root).

You are also required to write a main function which illustrates the usage of all member functions of the **`FullBinaryTree`** class.

Note: Follow these *good programming practices* when writing your code:

- There should be no memory leaks, dangling pointers, or any other type of runtime error in your program.
- Comment your code intelligently.
- Use meaningful variable and function names.
- Indent your code properly.
- Do not use any global or static variables.

☺ GOOD LUCK! ☺