LAB 8: PRIORITY QUEUES AND TRIES

F27SG – SOFTWARE DEVELOPMENT 3 (5 MARKS)

The topics of this lab are **priority queues** and **tries**. Download Lab8.zip from Vision and import the project into Eclipse. Your task in the lab is to complete the implementation and Junit tests for this project. Read the source code and make sure that you understand what is going on and what is missing. Before you start, you will need to download the ZIP file and import it into Eclipse:

File -> Import -> Existing Projects into Workspace

Then select the project you downloaded. The project is organized as follows:

- The src directory contains all the source files
  - PriorityQueue.java implements the priority queue for integers discussed in the lecture
  - **Q2.java** is where you should implement task 2.
  - **Trie.java** implements the Trie discussed in the lecture, and is where you should implement task 3 and the optional task 4.
- The test directory contains the unit tests for the project.

1. JUNIT TESTS [1 POINT]

Complete the JUnit test for the following Priority Queue operations in **PriorityQueueTest.java**:

- **insert** (check size and min)
- **removeMin** (check that elements are removed in correct order and size is correct)

Complete JUnit test for the `countAllWords` method in part 3. This method will count the number of words that has been inserted in to the Trie. This test should be completed in **TrieTest.java** and should test:

- for an empty trie
- for words that also contains “pre-fix” words (e.g. “ball” and “balloon”).

2. SORTING WITH PRIORITY QUEUE [2 POINTS]

In **Q2.java** there is a method

```
public static void sort(int[] arr){

}
```

Your task is to complete this, such that the method sorts the given array using a priority queue. **Q2Test.java** contains JUnit tests that should succeed, and **Q2.java** contains a main method so that you can run your program. When the main method is executed the numbers should be printed in order.

3. COUNTING THE NUMBER OF WORDS IN A TRIE [2 POINT]

You should complete the following method in **Trie.java**:

```
public int countAllWords(){

   // your code

}
```

This method should count the number of words stored in a Trie. Your implementation should be **recursive** and it is probably easiest to implement the recursion in the inner **TrieNode** class (and call this method from **countAllWords**)

Hint! There are other methods in the TrieNode class that perform a full traversal of the Trie and that you can use as a template to work from.

4. IS THERE A WORD WITH A GIVEN PREFIX? [OPTIONAL]

Given a prefix string, return true if and only if there is a string in the Trie that has that string as a prefix. E.g. true should be returned for the prefix “zeb” if a tree containing the word “zebra”. Again, it may be best to implement the helper method in **TrieNode recursively.** You should complete the following method (from **Trie.java**):

```
public boolean areWordsWithPrefix(String str){

}
```

Implement suitable JUnit tests for this method.