CSCI 2110 Computer Science III
Data Structures and Algorithms
ASSIGNMENT NO. 4

Date given: Saturday, November 18, 2017

Due: Monday, November 27, 2017 (11.55 p.m.)

The focus of this assignment is the heap data structure. You will be making simple changes to the Heap.java that was developed in the lectures.

Recall that a heap is essentially a priority queue. It is defined as a complete binary tree in which the key in every node x is greater than or equal to all the keys in the left and right subtrees of x. Since it is a complete binary tree, it can be implemented using a simple ArrayList that stores the keys in level order.

Download Heap.java and HeapDemo.java and understand the operations.

Part 1: Add the following methods to Heap.java

a) Method T findMin() that returns the key with the smallest priority. For instance, if the heap is

90	80	60	20	70	10	15	5	9	50
0	1	2	3	4	5	6	7	8	9

the method returns 5. As another example, if the heap is [20, 10, 15, 6, 9, 6], it should return 6. To implement this method, you must not search the entire arraylist. Rather, you should make use of the heap property and do a smart search. [Hint: the smallest element is always a leaf node].

b) Method T dequeueMin() that returns the key with the smallest priority and also deletes it. You can modify the sifting up operation to do this. For instance, if the heap is:

90	80	60	20	70	10	15	5	9	50
0	1	2	3	4	5	6	7	8	9

you would first find the smallest priority item using findMin(), then remove the last item (50) and put it in the place of 5.

90	80	60	20	70	10	15	50	9
0	1	2	3	4	5	6	7	8

Next sift 50 up until it finds the right spot.

90	80	60	50	70	10	15	20	9
0	1	2	3	4	5	6	7	8

If there are multiple keys with the minimum value, T dequeueMin() just removes one item.

Test the two methods in HeapDemo.java. Provide at least three sample outputs.

Part 2: Implement the HeapSort algorithm. The HeapSort algorithm is simple: “Build a heap and destroy it”. For this, write another client program HeapDemo2.java reads a text file with words, builds a heap with those words, and repeatedly removes the max item and puts it another text file. The resulting text file will have the words sorted in descending order. For example, if the input word file is:

```
there
their
about
would
these
other
which
```

the output word file is:

```
would
which
these
there
their
other
about
```

Test your program using a sample text file.

Part 3: Write a program HeapDemo3.java that has the following static method:

```
public static<T extends Comparable<T>> Heap<T> merge(Heap<T> heap1, Heap<T> heap2)
```

for merging two heaps. A simple way of merging two heaps is to copy the first heap into the result heap and then insert the items one by one from the second heap into the result heap. For instance, if the two heaps are:

90	80	60	20	70	10	15
0	1	2	3	4	5	6

85	20	70	10	5
0	1	2	3	4

the resulting merged heap will be:

90	85	60	80	70	10	15	20	20	70	10	5
0	1	2	3	4	5	6	7	8	9	10	11

Test your program for at least three sample outputs.
Bonus: The above strategy has an order of complexity O(nlogn). For bonus marks, can you design a faster algorithm?

What to submit: A zip file with Heap.java, HeapDemo1.java, HeapDemo2.java, HeapDemo3.java, sample inputs and outputs including the text files.