Computer Science III	Heap Lab: Words	Lab Value - 100

Lab Goal : This lab was designed to give you experience manipulating a heap by adding and removing with swap up and swap down, and to implement a complete heap sort.

Files Needed ::
`UnorderedWords.txt`
`WordHeapTester.java`

Lab Description : The file UnorderedWords.txt contains almost 10,000 words that are not in any particular order. The word should be read into an ArrayList based max heap. Instead of just ordering the words based on the alphabet, however, we will be ordering them based on the number of vowels. In order words, the value of a word will be the number of vowels (a, e, i, o, u) it contains. Words with more vowels should be at the top of the heap. This will be a max heap. If the vowel count of two words is the same, the one first in the alphabet should be considered larger, and thus should be removed first.

Write a Word class that has a String instance field, a compareTo method (so it's Comparable), a method to get the number of vowels in that word, and a method to get the word itself. It also needs a toString that includes the word and the number of vowels in some nice format so we don't have to count them! (see output).

Write a WordHeap class that has the ArrayList of Word objects. It will have an add method and a remove method that use the swap up and swap down algorithms we learned. You may add helper methods as desired, but you must write all of the heap functionality yourself. It also needs a toString that gives us a readable view of all the words (instead of all on one line with commas from ArrayList).

After successfully implementing the add and remove, write a complete heapSort that sorts a random array of words into order, ALPHABETICALLY this time, so use String compareTo instead of Word compareTo. You will need a fixHeap method which is very similar to the swap down used in remove.
For this section, add a constructor to WordsHeap that takes an ArrayList as a parameter.

What does add do?

Add adds a new Word in the very last spot in the tree(array spot length-1). Add then swaps up to restructure the tree so that the new item is in the proper location in the heap

What does remove do?

Remove copies the root to a variable so it can be returned and then moves the last value in the tree to the root-array spot 0. Remove then swaps down to restructure the tree and to check that the tree remains a heap.

What does heapSort do?

HeapSort first fixes each parent starting with the bottom-most parent so that each subtree is a max heap. It works its way backward through the array (ArrayList) fixing each parent until the root. At this point, the array should be a max heap.

The next step is the sorting. Each Word is removed and placed at the "end" of the ArrayList (after placing each one, the end becomes one index less). The last element is moved to the front/top, and the heap is fixed so that it is a max heap again.

Sample output with 20 words (test with MORE WORDS!):

Heap order:
automatically: 6
prettification: 6
countercharged: 5
topologies: 5
deontologists: 5
foreface: 4
pericardium: 5
hemodynamics: 4
destroyers: 3
monohulls: 3
refunders: 3
hiddenly: 2
monographic: 4
occasions: 4
haika: 3
tipi: 2
loup: 2
lotoses: 3
prevue: 3
obtests: 2

****

Removal order:
automatically: 6
prettification: 6
countercharged: 5
deontologists: 5
pericardium: 5
topologies: 5
foreface: 4
hemodynamics: 4
monographic: 4
occasions: 4
destroyers: 3
haika: 3
lotoses: 3
monohulls: 3
prevue: 3
refunders: 3
hiddenly: 2
loup: 2
obtests: 2
tipi: 2

****

Alphabetical order:
automatically: 6
countercharged: 5
deontologists: 5
destroyers: 3
foreface: 4
haika: 3
hemodynamics: 4
hiddenly: 2
lotoses: 3
loup: 2
monographic: 4
monohulls: 3
obtests: 2
occasions: 4
pericardium: 5
prettification: 6
prevue: 3
refunders: 3
tipi: 2
topologies: 5

Note that in heap order, the largest Word is in index 0 of the array. It's left and right children should be smaller. That pattern should hold through the array. If you're unsure, draw a picture.

In Removal order, the Words with the most vowels will be removed first. Within the same number of vowels, the Words should be removed in alphabetical order.