**13Programming Assignment**

Binary Search Tree and Heap – Word Frequency Counter

**Overview:**

You are given a code snippet to read from an input text file and produces an array of all words in that file.

You must implement a program that counts the frequency of words within an input file and output the word frequency count two different ways:

1. Ascending Order by word with the word and count output as such:

Word Count - Alphabetical

a, 20

apple, 1

arranged, 3

be, 3

badger, 1

…

1. Descending order by count with the word and count output as such:

Word Count - Frequency

A, 20

The, 18

It, 11

Java, 10

….

This approach can be used in activities such as natural language processing, development of a word dictionary from a text data source, implementation of search engines, etc.

**Directions:**

***General Requirements:***

* Given the sample code provided, implement a program that achieves the functionality discussed above.
* You will modify **WordCount.java**. A template with stubs is provided.
* Your program must utilize a **binary search tree** and a **max heap**.

***Goal #1 – Output in Ascending Order (alphabetically) by word:***

For your binary search tree, you must add each word with the:

* Key being the current word
* Info being the word’s count

Each time you add a word to a tree, you must first determine if the word is already present.

* If it is present, you must increment its count.
* If the word is not present, you must add it to the tree with an initial count = 1.

Once the full array is added to the binary search tree, you must output the binary search tree as an inorder traversal to the console.

This achieves requirement #1 from the overview.`3

***Goal #2 – Output in Descending Order by word count:***

You must output the keys in descending order based on word count (i.e. the most frequent words would be printed first). To accomplish this, I would suggest the following:

* Call Iteratable<key> it = bst.keys() to get an in-order (ascending) list of keys.
* Traverse the iterator, for each key and…
  + Look-up the count stored in the BST using the get method
  + Enqueue the key and value pair into the heap, but…
    - Use the value retrieved from the tree (the word frequency) as its priority 2
    - Use the key from the tree as the value.
    - i.e. we are storing the keys prioritized by their frequency.

Finally, you will need to dequeue and print every item in the queue to output the words in sorted order.

**Resources to use:**

* Binary Search Tree code as provided on the Canvas page or the author’s github site.
  + If you need to, you can change it.
* Heap code as provided on the Canvas page or the author’s github site.
  + If you need to, you can change it.

**Grading:**

|  |  |
| --- | --- |
| Output in Ascending Order   * Utilizes a binary search tree * Correctly implemented * Produces desired output | 45 |
| Output in Descending Order   * Utilizes a MaxHeap * Correctly implemented * Produces desired output | 45 |
| Code Style   * Commenting * Readability * Variable naming * References | 10 |
| Total | 100 |