**Requirements**

**1, Based on the provided Trie2.java source code, please write a Java method wordsPrefixedBy(TrieNode root, String p) in the provided Trie2 Prefix tree class, as described in below.**

**// First, please add this public method into the provided Trie2 class.**

**// Then implement the private helper method below this method.**

public LinkedList wordsPrefixedBy(String p) {

return wordsPrefixedBy(this.root, p);

}//end of method

// **The** **method returns a LinkedList of all words that have a prefix p. For example, if the current prefix**

**// tree object stores a set of words {apple, bike, bake, pen, did, ape, child, cat, file, hello, he, hell},**

**// the method call wordsPrefixedBy(root, “ap”) returns two words in the tree {“apple”, “ape”}**

**// Helper methods are allowed.**

private LinkedList wordsPrefixedBy(TrieNode root, String p) {

// **Write me**

//……..

}//end of method

2, Write a main() method in a separate **Tester.java** file, in which you will do the following,

2.1) Create a prefix tree instance named **myTrie**.

2.2) Insert into **myTrie** the following set of words, **{**apple, bike, bake, pen, did, ape, child, cat, file, hello, he, hell}, by repeatedly calling its insertString() method.

2.3) Your call **myTrie.wordsPrefixedBy(“ap”)** will return two words in the tree {“apple”, “ape”}. Please explicitly display on the standard output the list of words returned by your method call.

2.3) Your call **myTrie.wordsPrefixedBy(“he”)** will return three words in the tree {“hello”, “he”, “hell”}. Please explicitly display on the standard output the list of words returned by your method call.

3, Please organize your source code so that I can compile **all** your source files in **one** folder using command, **javac \*.java**, and run your program using command on command line, **java Tester.**

**4, You can have your own design for any details that have NOT been specified in this document. You have the freedom to add or change the code that I have provided.**