T9 Predictive Text

T9 is a predictive text technology for mobile featured phones (specifically those that contain a 4x3 numeric keypad). T9 stands for Text on 9 keys.

1	2 (abc)	3 (def)
4 (ghi)	5 (jkl)	6 (mno)
7 (pqrs)	8 (tuv)	9 (wxyz)
*	space	#

As you notice, there are keys labeled with digits 1–9, used for dialing phone numbers. These keys are also used to enter letters a–z. To type a text message the keys corresponding to the letters would be used. The user has to push each key multiple times for the desired char.Consider the word “hello”. Using this method, the user has to push 4, 4 to get char “h”, push 3, 3 to get char “e”, push 5, 5, 5 to get char “l” , push 5, 5, 5 to get char “l” , push 6, 6, 6 to get char “e”. For a 5 letter word it takes 14 button pushes which is extremely annoying.

To enter text more easily, the system of predictive text (also called “T9”) was devised. The user pushes each key only once and get the closest dictionary word that matches the word **(Auto Suggestions)**. The word “hello” can be typed in 5 button presses “43556” without pauses, instead of 14 in the standard system. The numeric string “43556” is referred to as a signature of the word “hello”. If this is the only match, the user can press space and carry on. If there are multiple matches, the user might need to select one of them before proceeding.

A given numeric-signature may correspond to more than one word. Predictive text technology is possible by restricting available words to those in a dictionary. Entering the numeric signature “4663” produces the words “gone” and “home” in many dictionaries.

In this test there are 5 steps for you to solve a basic t9 predictive text solution:

Step 1: Loading file to dictionary (1 mark)
A text file consists of SMS messages. Build a dictionary of these SMS messages where keys are words that occur in the SMS messages and values are frequencies of these words, i.e., the number of times a word occurs in all the messages. File path is “/Files/t9.csv”. To get the 1 mark input000.txt should pass. Write the code in loadDictionary method in Solution class**.** A helper method toReadFile(file) in Solution class is provided to you. It reads a file given as argument and returns a string array.

Step 2: Building a TST, Prefix operations (1 mark)
With the help of earlier dictionary, your task to load words in to TST with values as frequencies of words. To complete this task, you need to write code in T9 class constructor and getAllWords(String), by creating object of TST and inserting values as frequencies of word. T9 class constructor takes parameter BinarySearchST object. In getAllWords(String), you need to return keys that matches the given prefix. To get the 1 mark input001.txt should pass.

Step 3: Top K frequencies (1 mark)
To complete this test case, you need to write code in getSuggestions(Iterable, Integer) method. Method returns the top k(Integer) prefixes by combining all word possibilities based on their frequencies. If frequencies match, compare based on the word lengths. To get the 1 mark input003.txt should pass.

Step 4: Possible words for T9Signature word (1 mark)
To complete this step, your task is write code in potentialWords(String) which takes input as t9Signature string format i.e, 43556. Enumerate the words that are formed with the given T9 signature 43556.

To enumerate all words for given t9signature string, you need to create mapping for 4 as ‘g’, ‘h’, ‘i’, for 3 as ‘d’,’e’,’f’, for 5 as ‘’j’,’k’,’l’ and so on as per the figure below. Validate words by checking in the TST. To get the 1 mark input002.txt should pass. For example:
43556, words can be gdjjm, gdjjn, gdjjo, gdjkm, gdjkn,etc...

1	2 (abc)	3 (def)
4 (ghi)	5 (jkl)	6 (mno)
7 (pqrs)	8 (tuv)	9 (wxyz)
*	space	#

Step 5: Putting it all together (1 mark)
No need to fill code here. With the help of the all the methods working perfectly, it basically gives suggestions of the top k word possibilities based on the frequencies of a given t9signature word.
To get the 1 mark input004.txt should pass.