Practice #48: Spell-checker Dictionary

http://www.comp.nus.edu.sg/~cs1020/4_misc/practice.html

Objective:

- Using **HashMap** in Java

Task statement:

You are to write a program which performs spell-checking based on a spell-checker dictionary. In a spell-check dictionary, each entry consists of a misspelt word W and a list of correct words (C1, C2, C3, …) which can be misspelled as W due to one of the following reasons:

- One letter is deleted from a correct word (e.g., "reason" misspelled as "reson", "reaso").
- One letter is inserted into any position of a correct word (e.g., "reason" misspelled as "treason", "reatson")
- Two neighboring letters are transposed in the correct word (e.g., "reason" misspelled as "raeson", "reaosn")
- One letter in the correct word is replaced by a different letter in the alphabet (e.g., "reason" misspelled as "recson", "reasyn")

The input to your program is a list of correct words and a word to be checked. You program should construct a spell-checker dictionary from the list of correct words. Based on the list of correct words and the spell-checker dictionary, your program should give one of the following three responses for the word to be checked:

- An output of "OK" means that the word is in the list of correct words.
- An output of "Not found" means that the word is not in the list of correct words or the spell-checker dictionary.
- An output of "Possible corrections:", followed by the possible correct words (which may be misspelt into the given word due to the above-mentioned reasons) in alphabetical order.

You may assume that all the words are in lowercase, have at least 3 letters and do not contain any punctuation. You should implement your spell-checker dictionary using **HashMap**.

Input
The input consists of multiple lines. Line 1 contains an integer *K* and the next *K* lines each contains a distinct word. This is followed by a line that contains the word to be checked. All words are in lowercase letter with no punctuation marks.

(In your program, you should use more descriptive variable names instead of *K* and follow Java naming convention.)

Output
One of the three responses: "OK", "Not found", or "Possible corrections:" followed by a list of correct words in alphabetical order, each on a separate line.

Sample Input #1
```
2    // A list of 2 correct words
dog
cat
dog // Word to be checked
```

Sample output #1
```
OK
```

Sample Input #2
```
2      // A list of 2 correct words
dog
Cat
apple // Word to be checked
```

Sample output #2
```
Not found
```

Sample Input #3
```
2    // A list of 2 correct words
cat
can
cas // Word to be checked
```

Sample output #3
```
Possible corrections:
can
cat
```