**A Basic Spell Checker**

1. **Problem Overview**

This challenge will introduce you to the basics of Spell Checking. The objective is to build a solution that allows the detection and the correction of spelling mistakes in the English language. Most common spelling mistakes occur for the following reasons: Deletion, Replacement, Transposition and Insertion. We limit the scope of this project to alphabetical characters only.

**Deletion:** One character in the string is deleted incorrectly. Example: The user enters *Ordnary* instead of *Ordinary* (i.e. leaves out the i)

**Replacement:** One character in the string is incorrectly replaced by another one. Example: The user enters *Accedent* instead of *Accident*.

**Transposition:** The user swaps a pair of consecutive characters. Example: The user enters *Noramlly* instead of *Normally*.

**Insertion:** The user inserts one extra character somewhere in the string. Example: The user enters *Heello* instead of *Hello*. The extra character will only be a letter from [a-z] for the purpose of solving this problem.

So, generally, the correct string is just one step of one edit distance away from what the user erroneously types in. Given two strings str1 and str2, the edit distance (also called Levenshtein Distance) calculates the minimum number of edits required to convert str1 into str2. The operations that can be performed are insertion, deletion and replacement. Please take note, that in each of the four popular cases above, the mistake occurs only at one particular character (or, pair of characters in case 3). If a spell checker is able to detect these simple but common mistakes, it will be able to handle sixty to seventy percent of all spelling mistakes.

1. **Methodology**

You will be provided with a Corpus of text, which you can read in as a file in your program. Read in this text, and build up a dictionary of words and the frequencies with which those words occur. The text needs to be cleaned from all non-alphabetical characters except for hyphens and/or apostrophes, which might be part of a word. The end of the corpus file is marked by "END-OF-CORPUS"

For testing purposes, your program must allow the user to input a text file, correct it and save it. Your program should recommend the likeliest known word from the dictionary for each mistyped word. If the given word exists in your dictionary, output it as it is. The program must also print a report indicating the number of detected and corrected errors.

1. **Guidelines**
2. Consider the four popular mistakes described earlier in the description. Think of the candidate words which might have led to the given mis-typed versions.
3. Among the candidate words, restrict your choice to the words which do exist in your dictionary. Among these, pick up the word which occurred most frequently, as the best possible suggestion which you can find. If there are multiple such words then output the one which occurs first in lexicographical order.
4. Output should be in lower case.
5. The Corpus contains spelling errors as well. These errors can be considered as noise given that their statistical weight is not significant.
6. The use of Libraries is accepted **only** for individual steps of your approach. You cannot use an already existing function for spelling correction.
7. **Submission**

* Your code for dictionary construction and for spelling correction. **This challenge must be done using Python.**
* A four-page report that includes the following:
  + 1. A description of the methodology used for data cleaning.
    2. A discussion of the correction approach.
    3. Some experimental results.
    4. A discussion of the accuracy of the correction.
    5. A discussion of the limitations of your solution and possible improvements.