Text Editor Readme

1. **My Approach to the problem:** I have used Python 3 as my programming language for the project. Instead of using list data structure for making any edits in the text editor, I have used a data structure called **piece table**. The reason for choosing this data structure is that it allows to perform operations like insert, cut, copy, paste, and delete in O(1) time instead of O(n) time that occurs if list is used to store the document text. Besides this, I have also used **2 stacks**. One for storing the text after the undo operations and the other for storing the text after the redo operations. Piece table is a dictionary and my implementation of the piece table involves the following keys:

    a. "original" – It contains the original text. It is read-only i.e. it does not change no matter what operation is performed.
    b. "add" - It is an append only field where text is only appended at the end (takes O(1) time).
    c. "cut" – This key contains the text on the editor after performing the cut operation.
    d. "pieces" – It contains the piece descriptors i.e. objects of the class Pieces. Every piece descriptor is a part of the original text. Every piece contains three bits of information:
        i. source: tells us from which key to read text from
        ii. start: tells us from which index in the source text to read from
        iii. length: tells us how many characters to read from that text

    Various functionalities offered by my text editor are as follows:

    (a) delete text – takes 2 parameters offset and length and deletes the text of input length from the index i.e. the offset value. Finally, return the updated text.
    (b) undo text – pops the text present at the top of undo stack and appends it to the top of redo stack. Finally, return the updated text.
    (c) redo text - pops the text present at the top of redo stack and appends it to the top of undo stack. Finally, return the updated text.
    (d) highlight text – returns the text of input length starting from the offset index.
    (e) cut text – cuts the text of input length starting from the offset index.
    (f) copy text- copies the text of input length starting from the offset index.
    (g) paste text – pastes the cut text or copied text at the given offset index.
    (h) get text – returns the text currently on the editor.
    (i) misspellings checker – returns the count of misspelled words in the document text.

2. **Steps to run the program:** python editor.py
   Note: - I have included 2 additional files. First is **demofile.txt** from where I am reading the text and second is **spell.word.txt** that acts as a dictionary that contains ~50,000 english words. It is used for counting the number of misspelled words.
3. **Design decisions:** Besides using piece table, I have also used stack for implementing undo and redo operations as it provides O(1) time complexity for popping and pushing values into it.
4. **Extensions:**
   a. Besides the data structures that I have used, I would also like to implement splay trees in order to improve the performance because it provides quick access to recently accessed elements.
   b. Since there is not much difference in the benchmark of the original editor which used list data structure and my editor which uses piece table and stacks, I would also like to implement the data structures in C and expose the API in Python for speed.