Assignment 3

(Due: Dec 6, 2019) 11:59PM [NO EXTENSIONS WILL BE GRANTED]

Total points: 100

Introduction

In this assignment we will be spell checking large text files.

This assignment is to be completed **individually**. You are NOT allowed to share nor acquire source code from other students in the class, current or previous. You must start the assignment early in order to get proper help from the TAs and the instructor. You can also post general questions or ask for clarifications on Piazza, however, **do not post your own source code.**

Please read the entire assignment carefully, before starting

Reading Books

Your job is to write a program that will determine if the words in each of three different books are in the provided English dictonary. The script used to download the books can be found here. You can also find a link on the course webpage. The scriot can be run using the command given below (same to lab 10,11):

```
$source getbooks.sh
```

Note that the books are different than the ones used in the labs. You can find the dictionary needed for your work here. You can also find a link on the course webpage.

Creating a Hash Table

First, you must create a class which implements a hash table, using a hash function. This class, `HashTable` is made up of the following functions: `make_table()`, `make_hash()`, `insert()`, and `lookup()` (details on what these do are included in the code below). This class also has a constructor which takes in the length of the table, and a list of the same length which is made of strings. Your class must be written in the `assignment3.py` file.

```
class HashTable:
    """Hast Table Class"""
    def __init__(self, size: int = 0):
        """Constructor

        Takes a size, stores it
        Makes a list of the given size
        """
        self.size: int = size
        self.table: list = []
        self.make_table()

    def make_table(self) -> None:
        """Hash Table Creator

        Makes a table of the class's size"""

    def make_hash(self, word: str) -> int:
        """Hash Creator

        Takes a string, applies a hash method to it,
        and returns an int (the index to store the word)
        """

     def insert(self, word: str) -> None:
        """Insert

        Gets a hash for the given word, and
        attaches the word there
        """

    def lookup(self, word: str) -> bool:
        """Lookup

        Returns True if the word is in the table
        """
```

You must use this class to create your hash table.

Spell Check

Once you have downloaded the books, the dictionary, and implemented the `HashTable` class, read all of the words from the **dictionary** provided into your hash-table. The elements in the dictionary should be

converted to lowercase, before being inserted into your hash table. At this stage, the hash table is an unordered set.

> An unordered set is an associative container that contains a set of unique objects of type Key. Search, insertion, and removal can then be done on average in constant-time complexity. Internally, the elements of an unordered set are are not sorted in any particular order, but organized into buckets (hence hash-table). Which bucket an element is placed into depends entirely on the hash of its value. This allows fast access to individual elements, since once a hash is computed, it refers to the exact bucket the element is placed into.

Once you have loaded the dictionary, open a book (passed as a command line argument), and for each line:

1. Split the line into separate words (a 'word' in this case is a section of string with whitespace on either side)
2. Convert each word to lowercase, and clean up each word by removing non-alphanumeric values while retaining any apostrophe's (if this ends up turning that word into an empty string, that's okay)
3. Lookup each of these words from the book in your dictionary. Print out each word which is NOT in the dictionary, but is in the book.

Note

- **USE the exact method definitions as given for each question.** If you do not, then the autograder on Gradescope will not be able to recognize your answer, and will give you a zero.
- The hash table has to be implemented by you. You canot use the built-in collections/libraries provided by Python for this purpose.
    - ***If you are found to have used a built-in collection or library for the hash table, you will receive 0 points for this assignment***

Point Distribution

The sections below shows the distribution of points for the assignment:

Class Method	Points
`make_table()`	15 points
`make_hash()`	15 points
`insert()` and `lookup()`	30 points

Spell Checker	Points
Correct Output	40 points

SUBMISSION

You will submit ONE file called `assignment3.py` to Gradescope for this lab.

`assignment3.py` should contain your entire source code for this assignment.

Please provide meaningful comments and use proper coding style and indentation. There is no need to upload additional files. Feel free to create additional private/public methods, but you can't change/add data members.

Your program will be automatically graded. For each of the aforementioned tests you either pass the test case (full points) or not (zero points).

> We can test with any .txt file from the script.

Students caught cheating or plagiarizing will receive no credit. Additional actions, including a failing grade in the class or referring the case for disciplinary action, may also be taken.

Late submissions will receive a ZERO.

NOTE: Gradescope allows me to compare all submissions with each other and see how similar they are. If I find submissions which are too similar to each others, all the similar looking assignments will receive a ZERO. Disciplinary action might be taken depending upon the severity of the issue.

You are allowed *unlimited* resubmissions until the due date.