CS1110 Spring 2015

Assignment 6: Due Thursday April 16 at 11pm

You must work either on your own or with one partner. If you work with a partner, you and your partner must first register as a group in CMS and then submit your work as a group.

You may discuss background issues and general solution strategies with others, but the programs you submit must be the work of just you (and your partner). We assume that you are thoroughly familiar with the discussion of academic integrity that is on the course website. Any doubts that you have about "crossing the line" should be discussed with a member of the teaching staff before the deadline.

Objectives. Objects, lists of objects, functions and objects, dictionaries, and recursion. The assignment is based on Lectures 19 and 20 (with reenforcement from Lectures 21 and 22) and Labs 8 and 9. It is essential that you begin this assignment early.

1 Set-Up

Set up a folder `A6` by unzipping the file `A6.zip` that can be downloaded from the assignments page. It includes (a) a template module `Sonnets.py` that you are to develop and ultimately submit to CMS, (b) a module of helper functions `SonnetTools.py`, (c) a data file `TheSonnets.txt`, and (d) a collection of test modules that can be used for debugging. It is not a bad idea to browse through all this stuff before you begin, reading the doc strings etc. LET A6 BE THE CURRENT WORKING DIRECTORY WHENEVER YOU ARE WORKING ON THIS ASSIGNMENT. Make sure that everything that is in the zip file is in this working directory.

2 Background

Shakespeare wrote 154 sonnets. Here is one of the more famous:

Shall I compare thee to a summer's day?
Thou art more lovely and more temperate:
Rough winds do shake the darling buds of May,
And summer's lease hath all too short a date:
Sometime too hot the eye of heaven shines,
And often is his gold complexion dimmed,
And every fair from fair sometime declines,
By chance, or nature's changing course untrimmed:
But thy eternal summer shall not fade,
Nor lose possession of that fair thou ow'st,
Nor shall death brag thou wander'st in his shade,
When in eternal lines to time thou grow'st,
  So long as men can breathe, or eyes can see,
  So long lives this, and this gives life to thee.

A Shakespearean sonnet (hereafter just "sonnet") has 14 lines with rhyming pattern

A-B-A-B-C-D-C-D-E-F-E-F-G-G.

Each line has ten syllables.

In this assignment you are provided with a text file that contains all the sonnets and tools for reading the data in that file into your Python environment. You will develop a Python class called `Sonnet` and a number of functions that work with `Sonnet` objects. Upon completion you will have code that can (a) "pretty print" a sonnet and report its word count and index, (b) sort a list of sonnet objects based on first lines, (c) "modernize" a sonnet by translating such words as "thou" and "dost" into "you" and "do", and (d) create a rhyming dictionary based on all the rhyming pairs that show up in the sonnet collection. *It is essential that you begin this assignment early.*

3 The Sonnet Class

In the tempate module `Sonnets.py` you will see specifications for the class `Sonnet` that you are to develop. The class has five attributes:

`index`	`a string that encodes the sonnet number as a Roman Numeral`
`text`	`a length-14 list of strings that encodes the sonnet`
`firstLine`	`a string that is the first line of the sonnet without punctuation`
`nWords`	`an int that is the number of words in the sonnet`
`lastWords`	`a length-14 list of strings, one for the last word in each line`

Your first task is to complete the constructor function `__init__(self,index,text)`.

The module `SonnetTools.py` contains two functions that can be used to facilitate the computation of the `firstLine` and `lastWords` attributes. The function `dePunc(s)` can be used to remove all punctuation from a string and the function `lastWord(s)` can be used to extract the last "word" in a string. Regarding `nWords`, you may assume that the number of words in a sonnet line equals one more than the number of blanks in the sonnet line. The module `TestA.py` can be use do check your implementation of the constructor `__init__(self,index,text)`.

4 Creating a List of Sonnet Objects

Take a look at the file `TheSonnets.txt` that contains 152 of the original the 154 sonnets. (For simplicity in this assignment, we have deleted Sonnet XCIX which has 15 lines and Sonnet CXXVI which has twelve lines.) The information for a particular sonnet is housed in 17 contiguous lines. For example, if you visit that part of `TheSonnets.txt` that houses Sonnet CVIII you will discover that things are laid out this way:

```
XVIII.                                                <--- Line 0  for this sonnet
                                                      <--- Line 1  for this sonnet
Shall I compare thee to a summer's day?               <--- Line 2  for this sonnet
Thou art more lovely and more temperate:              <--- Line 3  for this sonnet
Rough winds do shake the darling buds of May,         <--- Line 4  for this sonnet
And summer's lease hath all too short a date:         <--- Line 5  for this sonnet
Sometime too hot the eye of heaven shines,            <--- Line 6  for this sonnet
And often is his gold complexion dimmed,              <--- Line 7  for this sonnet
And every fair from fair sometime declines,           <--- Line 8  for this sonnet
By chance, or nature's changing course untrimmed:     <--- Line 9  for this sonnet
But thy eternal summer shall not fade,                <--- Line 10 for this sonnet
Nor lose possession of that fair thou ow'st,          <--- Line 11 for this sonnet
Nor shall death brag thou wander'st in his shade,     <--- Line 12 for this sonnet
When in eternal lines to time thou grow'st,           <--- Line 13 for this sonnet
So long as men can breathe, or eyes can see,          <--- Line 14 for this sonnet
So long lives this, and this gives life to thee.      <--- Line 15 for this sonnet
                                                      <--- Line 16 for this sonnet
```

Line 0 contains the sonnet's index as a Roman numeral. Lines 1 and 16 are blank. Lines 2 through 15 contain the sonnet text. The function `GetSonnets()` assembles all this data into a length-2584 list of strings. (Note: $2584 = 152 \times 17$). Run the test module `ShowGetSonnets` for further insight.

Implement a function `MakeSonnetList(L)` that returns a length-152 list of sonnet **objects** where `L` is a list of strings as produced by `GetSonnets()`. Thus, if we execute

```
L = GetSonnets()
SonnetList = MakeSonnetList(L)
printSonnet(SonnetList[17])
```

then information about the sonnet that `SonnetList[17]` encodes would be displayed. (Note: `SonnetList[17]` encodes sonnet XVIII (=18).)

The test modules `TestB1.py`, `TestB2.py`, and `TestB3.py` can be used to confirm the correctness of your work on this part of the assignment. The function `MakeSonnetList` should be part of the finished module `Sonnets.py` that you submit to CMS.

5 Modernizing a Sonnet

The sonnets are written in what is called Early Modern English. While not really confusing, we would like to convert some of the vocabulary to Modern English as a courtesy to the I'm-too-busy-texting reader! We will only pay attention to a few common words:

Early Modern English	Modern English
thou	you
thy	your
thee	you
thine	your
art	are
doth	do
hast	have
hath	has

Early Modern English	Modern English
Thou	You
Thy	Your
Thee	You
Thine	Your
Art	Are
Doth	Do
Hast	Have
Hath	Has

We say that a sonnet is *modernized* if every occurrence of one of these Early Modern English words is replaced by its Modern English equivalent. Thus, the modernized version of Sonnet XVIII (given above) is

```
Shall I compare you to a summer's day?
You are more lovely and more temperate:
Rough winds do shake the darling buds of May,
And summer's lease has all too short a date:
Sometime too hot the eye of heaven shines,
And often is his gold complexion dimmed,
And every fair from fair sometime declines,
By chance, or nature's changing course untrimmed:
But your eternal summer shall not fade,
Nor lose possession of that fair you ow'st,
Nor shall death brag you wander'st in his shade,
When in eternal lines to time you grow'st,
   So long as men can breathe, or eyes can see,
   So long lives this, and this gives life to you.
```

Implement a function `Modernize(S)` that takes a sonnet object `S` and returns a sonnet object `T` with the property that `T.text` encodes the modernized version of the sonnet encoded by `S.text`. A template for `Modernize` is provided in the module `Sonnets.py`. Notice that we have provided a dictionary to facilitate the process. To receive full credit, you must make effective use of this dictionary. Your implementation will require a nested loop. Each line in the sonnet must be scanned for occurrences of each key in the dictionary. Any time an occurrence is found, it must be replaced by the value associated with the key. Thus, a line like

```
'Thou art a smart CS 1110 student who hast tons of time to spend in lab.'
```

becomes

```
'You are a smart CS 1110 student who has tons of time to spend in lab.'
```

We have set up the dictionary to minimize the chance of "accidental" substitutions. For example, we wouldn't want to convert `'smart'` to `'smare'`!

The test module `TestC.py` can be used to check out your implementation of `Modernize`. This function must be part of the finished version of `Sonnet.py` that you are to submit to CMS.

6 Rhyming Dictionaries

We define a *rhyming dictionary* as a Python dictionary whose keys are strings and whose values are lists of distinct strings. Thus

```
D1 = {’x’:[’a’,’b’,’c’],y:[’g’,’h’]}
```

is a rhyming dictionary while

```
D2 = {’x’:[’a’,’b’,’a’],y:[’g’,’h’]}
```

is not because `[’a’,’b’,’a’]` has a repeat value. When building a rhyming dictionary, the order of the values in the lists is unimportant. Thus, `[’a’,’b’,’c’]` is just as good as `[’c’,’a’,’b’]`.

Suppose you have a list of sonnet objects `SL`. Think of all the “last words” in those sonnets. You are going to construct a rhyming dictionary `D` whose keys are “last word” strings and whose values are lists of last word strings. In particular, if we have an item in `D` like

```
’day’ : [’may’,’sway’,’lay’]
```

then this will mean that across the collection of sonnets encoded in `SL`, “day” is rhymed with “may” in at least one sonnet, “day” is rhymed with “sway” in at least one sonnet, and “day” is rhymed with “lay” in at least one sonnet. If `SL` encodes the entire collection of 152 sonnets, then by looking at `D` we can see how the Bard chose to rhyme what with what.

Let’s clarify what we mean by “one word is rhymed with another” in a sonnet. It gets back to the A-B-A-B-C-D-C-D-E-F-E-F-G-G rhyming pattern. Using Sonnet XVIII (again) as an example we see

day		may	Shall I compare thee to a summer’s day?
temperate		date	Thou art more lovely and more temperate:
may		day	Rough winds do shake the darling buds of May,
date		temperate	And summer’s lease hath all too short a date:
shines		declines	Sometime too hot the eye of heaven shines,
dimm’d		untrimm’d	And often is his gold complexion dimmed,
declines	rhymes with	shines	And every fair from fair sometime declines,
untrimm’d		dimm’d	By chance, or nature’s changing course untrimmed:
fade		shade	But thy eternal summer shall not fade,
ow’st		grow’st	Nor lose possession of that fair thou ow’st,
shade		fade	Nor shall death brag thou wander’st in his shade,
grow’st		ow’st	When in eternal lines to time thou grow’st,
see		thee	So long as men can breathe, or eyes can see,
thee		see	So long lives this, and this gives life to thee.

There are three steps to building a rhyming dictionary for a list of sonnet objects and they each require the implementation of a function. These will now be described.

7 “Adding” a Pair of Rhyming Dictionaries

Here are two rhyming dictionaries:

```
D1 = {’happy’:[’C’,’D’,’B’,’A’],’sad’:[’X’,’Y’,’Z’]}
D2 = {’happy’:[’F’,’G’,’D’,’C’],’blah’:[’W’,’X’]}
```

and here is their “addition”:

```
D = {’happy’:[’C’,’D’,’B’,’A’,’F’,’G’],’sad’:[’X’,’Y’,’Z’],’blah’:[’W’,’X’]}
```

(Remember, the order of the values in the lists is not important.) Here is a pseudocode description of the algorithm for creating `D` from `D1` and `D2`:

```
D1 = copy.deepcopy(D1)
D2 = copy.deepcopy(D2)
D = D1
for each key d in D2 do the following:
    If d is not a key in D, then make  d:D2[d] an item in D. Otherwise, append
    to D[d] each value v in D2[d] that is not in D[d].
```

When deciphering this, remember that the keys are strings and the values are lists of distinct strings.

Given this definition of what it means to add two rhyming dictionaries, complete the implementation of the following function so that it performs as specified.

```
def Add(D1,D2):
    """ Returns a dictionary that is the sum of D1 and D2

    D1 and D2 are not modified

    PreC: D1 and D2 are rhyming dictionaries
    """
```

Some hints. (1) Avoid subtle referencing errors by making deep copies of D1 and D2 as suggested by the pseudocode above. (2) Make sure your implementation can handle the situation when either D1 or D2 is the empty dictionary.

Your implementation of Add should be part of the module Sonnets.py that you are to submit to CMS. The module TestD.py may be of interest as you debug your implementation of Add.

8 A Rhyming Dictionary for a Single Sonnet

The function Add makes it easy to create a rhyming dictionary for a single sonnet. Suppose S is a sonnet object and L = S.lastWords. Following the discussion in §6 we simply use Add to "add up" the 14 "baby" rhyming dictionaries that are based on the A-B-A-B-C-D-C-D-E-F-E-F-G-G rhyming pattern:

```
{L[0]:[L[2]]}
{L[2]:[L[0]]}
{L[1]:[L[3]]}
{L[3]:[L[1]]}
     etc
```

Complete the following function so that it performs as specified:

```
def SonnetDict(S):
    """ Returns a rhyming dictionary based on the
    last words in the sonnet encoded in S.

    S is not modified.

    PreC: S is a sonnet object.
    """
```

The modules TestE1.py and TestE2.py may be of interest as you debug this part of the assignment. Observe that the rhyming dictionary for some sonnets have length less than 14. That is because some sonnets have fewer than 14 distinct last words. (Check SonnetList[5].)

Your implemention of SonnetDict should be part of the finished module Sonnets.py that you are to submit to CMS.

9 A Rhyming Dictionary for a List of Sonnet Objects

At last we are ready to compute a rhyming dictionary for a list of sonnet objects `SL`. One way is through a repeated addition process:

```
D = dict()
for each sonnet S in the sonnet list SL do the following:
   Update D by adding to it the rhyming dictionary for S.
```

Use this pseudocode to develop a nonrecursive implementation of the following function:

```
def MakeRhymeDict(SL):
    """ Returns a rhyming dictionary associated with the sonnet list SL.

    SL is not modified

    PreC: SL is a list of sonnet objects.
    """
```

The module `TestF1.py` can be used to affirm the correctness if your implementation.

There is an interesting recursive alternative to `MakeRhymeDict` and it is very similar in structure to recursive merge sort. If `SL` is a list of sonnet objects, then we can recur as follows:

```
If SL has length 1, then return the rhyming dictionary of the single
      sonnet that it encodes.
Otherwise, (a) split SL into a pair of (approximately) half-sized sonnet
      lists S1 and S2, (b) compute their respective rhyming dictionaries D1
      and D2, and (c) return the addition of D1 and D2.
```

Develop a recursive function `MakeRhymeDictR(SL)` along these lines. It should have exactly the same specification as `MakeRhymeDict(SL)`. The module `TestF2.py` can be used to affirm the correctness of your implementation.

The functions `MakeRhymeDict` and `MakeRhymeDictR` should be part of the finished module `Sonnets.py` that you are to submit to CMS.

10 Summary and Scoring

You are to submit a single module `Sonnets.py` to CMS. It should have your implementations of the following functions and nothing else:

```
__init__(self,index,text)      #  6 points
MakeSonnetList(L)              # 10 points
Modernize(S)                   #  6 points
Add(D1,D2)                     # 10 points
SonnetDict(S)                  #  6 points
MakeRhymeDict(SL)              #  6 points
MakeRhymeDictR(SL)             #  6 points
```