CSCI 1133, Fall 2019
Programming Assignment 9
Due: 11:55pm, Wednesday November 6, 2019

Due Date: Submit your solutions to GitHub by 11:55 p.m., Wednesday, November 6th. We will do a pull from this time point. Do not upload anything to Canvas and PLEASE be sure to use proper naming conventions for the file, classes, and functions. We will NOT change anything to run it using our scripts.

Unlike the computer lab exercises, this is not a collaborative assignment. You must design, implement, and test your code on your own without the assistance of anyone other than the course instructor or TAs. In addition, you may not include solutions or portions of solutions obtained from any source other than those provided in class (so you are ONLY allowed to reuse examples from the textbook, lectures, or code you and your partner write to solve lab problems). Otherwise obtaining or providing solutions to any homework problem for this class is considered Academic Misconduct. See the syllabus and read section "Academic Dishonesty" for information concerning cheating. Always feel free to ask the instructor or the TAs if you are unsure of something. They will be more than glad to answer any questions that you have. We want you to be successful and learn so give us the chance to help you.

Instructions: This assignment consists of 2 problems, worth a total of 40 points. Solve the problems below by yourself, and put all functions in a single file called `hw9.py`. Use the signatures given for each class and function. We will be calling your functions with our test cases so you must use the information provided. If you have questions, ask!

Because homework files are submitted and tested electronically, the following are very important:

- You follow all naming conventions mentioned in this homework description.
- You submit the correct file, hw9.py, through Github by the due date deadline.
- You follow the example input and output formats shown.
- Regardless of how or where you develop your solutions, your programs should execute using the python3 command on CSELabs computers running the Linux operating system.

Push your work into Github under your own repo. The specific hosting directory should be: repo-<username>/hw9, where you replace <username> with your U of M user name. For instance, if your email address is bondx007@umn.edu, you should push your hw9 to this directory: repo-bondx007/hw9

The following will result in a score reduction equal to a percentage of the total possible points:

- Incorrectly named/submitted source file, functions, or classes (20%)
- Constraints not followed (40%)
- Failure to execute due to syntax errors (30%)

Use the following template for EVERY function that you write (note: a helper function is a function so it gets a template.)

```
#========================================
# Purpose: (What does the function do?)
# Input Parameter(s): (Each parameter by name and what it represents)
# Return Value(s): (What gets returned? Possibilities?)
#========================================
```

Problem A. (20 *points*) **Writing Really Bad Fanfiction**

Getting computers to produce convincing English sentences about a particular topic is a challenging problem, so we won't be doing that in this assignment. Instead, we'll use a shortcut that quickly produces nonsense that somewhat resembles coherent English. Given a sequence of sentences on a topic (for example, a novel), our strategy to produce a sentence will be as follows:

- For each distinct word in the novel, determine how frequently any other given word follows it, including how frequently nothing follows it (because the sentence ends). Note that capitalization matters here: we are counting 'Hello' and 'hello' as distinct words (this will help ensure that our sentences usually start with capitalized words).
- To start generating a sentence, pick a word at random from any of the words that start a sentence in the novel.
- To generate the next word, pick one of the words that followed the current one at random (in a weighted format, so if word A is followed by word B twice as often as word C in the novel, then word A is twice as likely to be followed by word B than word C in our sentence). This includes the possibility of picking nothing.
- Keep choosing words until nothing is picked, at which point the sentence ends.

To do this, you need to write three functions (the first two are helper functions for the last one). Be sure you match the names and parameters of the functions exactly to prevent grading issues.

`first_words(fname)` takes in fname, a string representing the name of a file, and returns a list of the first word in every sentence in that file, in order, including duplicates. See below for specifications on the formatting of the input file. You can assume that the file specified actually exists (i.e. you don't have to worry about handling FileNotFoundError on this problem).

`next_words(fname)` takes in fname, a string representing the name of a file, and returns a dictionary where the keys are each distinct word in the file (case matters), and the value for any given key is a list of every word that follows that key anywhere in the file, in order, including duplicates. In the case that a given key appears as the last word of a sentence, you should add the string `"."` to the list instead of the word that follows it.

`fanfic(fname)` takes in fname, a string representing the name of a file, and prints 10 'sentences', one per line, based on that file, as described above. In particular, the first word of the sentence should be randomly chosen from the list produced by `first_words` (use `random.choice`, which chooses an element at random from a list). Each word produced should then be used as a key into the dictionary produced by `next_words`, and then the next word should be chosen at random from the list that is the

value for that key. The sentence should end when `"."` is chosen. The function should not return anything.

You can make the following assumptions about the input files (see the examples provided on Canvas):

- They exist in the directory the script is being run from, and are readable text files.
- Every line will be a single sentence, ending with a period (separated from the last word by a space). Each word in a given sentence will be separated by a single space.
- There will be no other punctuation in the file, aside from apostrophes used in contractions.

Constraints:

- Do not import/use any Python modules other than random
- You may use any string or list method that is appropriate to solving this problem.
- Don't use the input() function, as this will break our grading scripts.
- Your submission should have no code outside of the function definitions, aside from comments and import random.

Examples (download the sample text files from Canvas into the folder you're running hw9.py from):

```
>>> first_words('short1.txt')
['Never', 'Never', 'Never']

>>> first_words('short2.txt')
['Fear', 'Fear', 'Anger', 'Hate', 'I']
```

(note: remember that order does not matter for dictionaries, so you may get the keys in a different order in the next two examples, but the key-value pairings should be the same)

```
>>> next_words('short1.txt')
{'let': ['you'], 'down': ['.'], 'Never': ['gonna', 'gonna', 'gonna'],
'gonna': ['give', 'let', 'run'], 'and': ['desert'], 'up': ['.'],
'run': ['around'], 'you': ['up', 'down', '.'], 'around': ['and'],
'give': ['you'], 'desert': ['you']}

>>> next_words('short2.txt')
{'anger': ['.'], 'dark': ['side'], 'I': ['sense'], 'the': ['path',
'dark'], 'you': ['.'], 'Anger': ['leads'], 'much': ['fear'], 'Hate':
['leads'], 'fear': ['in'], 'sense': ['much'], 'to': ['the', 'anger',
'hate', 'suffering'], 'side': ['.'], 'leads': ['to', 'to', 'to'],
'Fear': ['is', 'leads'], 'path': ['to'], 'is': ['the'], 'hate': ['.'],
'in': ['you'], 'suffering': ['.']}
```

(note: the output of `fanfic` is random, so these test cases only demonstrate the format of the output: your actual sentences will not match)

```
>>> fanfic('alice_in_wonderland.txt')
```

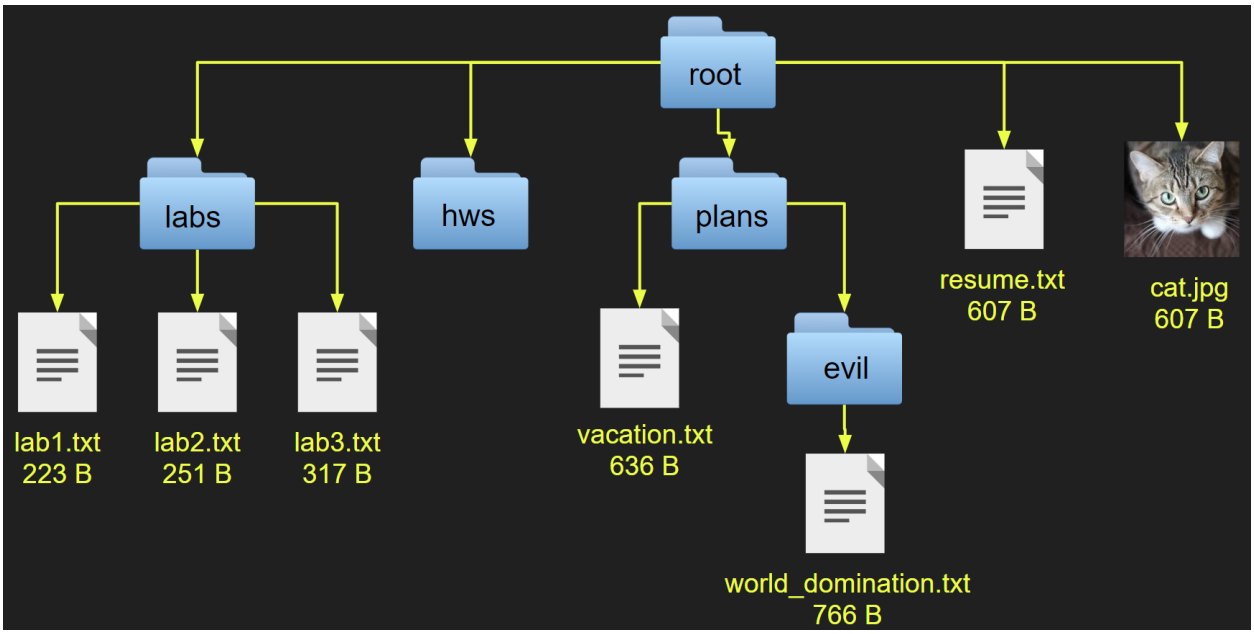
I only been the Hatter .
Keep back by the best .
Come away .
While the second thing she crossed her head pressing against the Hatter and looking party went on half shut his eyes then said Alice thought .
The Hatter continued in despair she felt quite a vegetable .
If you see it really you are the Caterpillar and get through that he began in livery came very like an air .
Then I'll write this be no use of the question but she put on just as the simple question said the air Do you begin lessons you'd better leave out to the right said Alice who is .
First because they're only as they don't fit .
The Duchess she waited to herself and after the fan and the other guests mostly Kings and very slowly back by the White Rabbit Hole .
So Bill's got altered .

```
>>> fanfic('wizard_of_oz.txt')
```
So he had they were waiting in a slate from the room .
She sprang into the most noble Sorceress to Kansas again I know it gently on he was .
Dorothy said and sharp edge of the Woodman's weapon .
So they fluttered in a great sorrow was to the silk into her .
The Lion she was frightened when the daylight the other in the Tin Woodman to forgive us answered the green .
These tears of the Scarecrow .
We must run away through the Scarecrow twisted its body was so brightly in the huge beast .
I got the King of towers and you not know what made of the magic words the animals as he wishes .
The fences and as I may thereafter be dreadfully worried Dorothy heard a plate to help to guard against mishap .
They now came to fool me back so she asked Is he said to an instant the china fences at him a heart beat fast asleep .

Problem B. (20 *points*) **Directory Traversal**

One method for representing a simple file system is with nested dictionaries, where each dictionary represents a directory. Each key represents the name of either a file or a subdirectory, and each value is either another dictionary for a subdirectory, or metadata for a file. In this case, we'll be looking at memory usage, so the value for each file key will be the size of the file in bytes. For example, we could represent the directory root in the image below as the following dictionary:

```
root = {'labs':{'lab1.txt':223,
                'lab2.txt':251,
                'lab3.txt':317,},
        'hws':{},
        'plans':{'vacation.txt':636,
                 'evil':{'world_domination.txt':766}},
        'resume.txt':607,
        'cat.jpg':607}
```



Write a function `total_txt_size(directory)` that takes in a nested dictionary representing a directory as explained above, and returns the total memory in bytes (an integer) being used by `txt` files in the directory (that is, any file where the last four characters in the filename are `'.txt'`).

Hints:

- You need to sum the memory of all `.txt` files within the directory at any level within the hierarchy, so this includes files in the root directory, files in subdirectories of the root, files in subdirectories of those subdirectories, and so on.
- You're not required to use recursion, but it will probably make the solution much simpler. This is an example of tree traversal, a problem type that is particularly suited to recursive solutions.

- To check whether a given element in the dictionary represents a file or a subdirectory, check whether the value in the key-value pair is type `int`.

Constraints:

- Do not import/use any Python modules (except random for part A)
- Don't use the input() function, as this will break our grading scripts.
- You may use any string or list method that is appropriate to solving this problem.
- Your submission should have no code outside of the function definitions, aside from comments and import random.

Examples:

```
>>> root1 = {'time_travel':{'micro_black_holes.zip':100,
                            'quantum_realm.pdf':150,
                            'magic.txt':200},
          'space_travel':{'tesseract.java':400,
                           'magic.txt':100,
                           'warp_drive.py':300,
                           'mass_relay.txt':100}}
>>> total_txt_size(root1)
400

>>> root2 = {}
>>> total_txt_size(root2)
0

>>> root3 = {'labs':{'lab1.txt':223,
                     'lab2.txt':251,
                     'lab3.txt':317,},
            'hws':{},
            'plans':{'vacation.txt':636,
                     'evil':{'world_domination.txt':766}},
            'resume.txt':607,
            'cat.jpg':607}
>>> total_txt_size(root3)
2800

>>> root4 = {'a':{'b':{'c':{'d':{'e':{'f.c':10}},'g.txt':12}}}}
>>> total_txt_size(root4)
12
```