CMPSC-132: Programming and Computation II

Lab #1

Due Date: 01/21/2022, 11:59PM
10 pts

General instructions:

- The work in this assignment must be your own original work and be completed alone.
- The instructor and course assistants are available on Teams and with office hours to answer any questions you may have. You may also share testing code on Teams.
- A doctest is provided to ensure basic functionality and may not be representative of the full range of test cases we will be checking. Further testing is your responsibility.
- Debugging code is also your responsibility.
- You may submit more than once before the deadline; only the latest submission will be graded.

Assignment-specific instructions:

- Download the starter code file from Canvas. Do not change the function names or given starter code in your script.
- Each question has different requirements, read them carefully and ask questions if you need clarification. No credit is given for code that does not follow directions.
- Watch the Module 1 lectures before attempting the assignment.
- If you are unable to complete a function, use the pass statement to avoid indentation errors.
- Your file must contain only the function's code. Testing and debugging code must be removed

Submission format:

- Submit your LAB1.py file to the Lab 1 Gradescope assignment before the due date.
- As a reminder, code submitted with syntax errors does not receive credit, please run your file before submitting.

isValid(a_string) **(2 pts)**

Takes a string *a_string* and returns True if *a_string* is a string that has exactly 26 characters and each of the letters 'a'/'A' -'z' /'Z' appeared only once in either lower case or upper case, False otherwise.

Preconditions:

You should not make any assumptions about the size or contents of *a_string*.

Input	
a_string	A Python string

Output	
bool	True if *txt* uniquely contains all the characters of the English alphabet, False otherwise

Example:

```
>>> isValid('qwertyuiopASDFGHJKLzxcvbnm')
True
>>> isValid('hello there, spring is here!')
False
>>> isValid('POIUYTqwerASDFGHlkjZXCVBMn')
True
```

Length Distribution **(3 pts)**

In this problem, we will test your work in 2 ways: (i) as standalone functions, get_words (0.5 pts) and get_histogram (1.5 points); and (ii) correct functionality when sending returned values from get_words to get_histogram (1 point).

get_words(filename)

Given a *filename* as a string, complete the implementation of this function given in the starter code to open the file and retrieve words from each line to create (and return) a list with whitespaces stripped.

Hints:

- The str.strip() method returns a copy of the string with the leading and trailing characters removed, whitespace by default.

Input	
filename	A valid file name (as a Python string) of a .txt file saved in working directory

Output	
list	A list with the sequence of all the words contained in *filename*

Example:

```
.txt file for this doctest is available on Canvas and must be saved in the
same directory as your .py file
>>> get_words('contents.txt')
['week', 'bat', 'aquatic', 'eggs', 'threatening', 'crash', 'educated',
'adjoining', 'bent', 'mice', 'belief', 'adjustment', 'blood', 'smooth',
'kaput', 'mountain', 'digestion', 'enchanted', 'wandering', 'fresh']
```

get_histogram(words)

Takes a Python list *words* and returns a dictionary where the key is the length of a word and the value is the count of how many words in the list are of that length, in other words, a histogram dictionary mapping length to word counts.

Preconditions:

While the function will receive lists, you should not make any assumptions about the size of *words*.

Input	
words	A Python list

Output	
dict	A dictionary mapping length to word counts

Example:

```
>>> get_histogram(['hello', 'there', 'spring', 'is', 'here'])
{5: 2, 6: 1, 2: 1, 4: 1}
>>> list_of_words = get_words('contents.txt')
>>> get_histogram(list_of_words)
{4: 4, 3: 1, 7: 1, 11: 1, 5: 4, 8: 2, 9: 4, 6: 2, 10: 1}
```

removePunctuation(a_string) **(5 pts)**

Takes a non-empty string *a_string* and returns 2 values. The first returned value is a string where each character in *a_string* that *is not* an alphabet letter is replaced with a space. The second returned value is a dictionary with the frequency count of each character removed from *a_string*, where the key is the character, and the value, the number of times the character was removed from the string.

Hints:

- The str.isalpha() method returns True if all characters in a string are alphabet letters (a-z).
- When x = 'We', doing list(x) produces the list ['W', 'e']

Preconditions:

While the function will receive non-empty strings, you should not make any assumptions about the characters in *a_string*

Input	
a_string	A non-empty string

Output	
str, dict	First value is a copy of aString where each non-alphabet letter is replaced with a space. Second value is a dictionary with the frequency count of each character removed from *aString*. Ignore space count

Example:

```
>>> removePunctuation("Dots..................... many dots..X")
('Dots                        many dots  X', {'.': 24})
>>> data = removePunctuation("I like chocolate cake!!(!! It's the best
flavor..;.$ for real")
>>> data[0]
'I like chocolate cake      It s the best flavor      for real'
>>> data[1]
{'!': 4, '(': 1, "'": 1, '.': 3, ';': 1, '$': 1}
# Don’t forget dictionaries are unsorted collections, key order does not
matter
```