CMPSC-132: Programming and Computation II

Lab #1

Due Date: 09/03/2021, 11:59PM EST
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 syntax 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.

joinList(n) (2 pts)

Takes an integer number *n*. This functions returns the list [1,2,...,*n*,*n*,...2,1] for every *n*>0.

Preconditions:

While the function will receive integer values, you should not make any assumptions about the value of *n*.

Input	
n	An integer value

Output	
list	Starting at 1 moving upwards, a list with the sequence 1 to *n*, followed the sequence that starts at n moving downwards to 1
None	Argument do not satisfy the conditions of the function

Example:

```
>>> joinList(5)
[1, 2, 3, 4, 5, 5, 4, 3, 2, 1]
>>> joinList(1)
[1, 1]
>>> joinList(-3) is None
True
```

isValid(txt) **(3 pts)**

Takes a string *txt*, and returns True if *txt* is a string that has 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 *txt*.

Input	
txt	A string

Output	
bool	True if *txt* uniquely contains all the characters of the English alphabet, False otherwise
None	Arguments do not satisfy the conditions of the function

Example:

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

removePunctuation(aString) **(5 pts)**

Takes a non-empty string *aString* and returns 2 values. The first returned value is a string where each character in *aString* 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 *aString*, 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 *aString*

Input	
aString	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
```