Homework 4 - Strings, Indexing, and Lists

CS 1301 - Intro to Computing - Fall 2021

Important

- Due Date: **Tuesday, September 28th, 11:59 PM**.
- This is an individual assignment. High-level collaboration is encouraged, **but your submission must be uniquely yours.**
- Resources:
  - TA Helpdesk
  - Email TA's or use class Piazza
  - How to Think Like a Computer Scientist
  - CS 1301 YouTube Channel
- Comment out or delete all function calls. Only import statements, global variables, and comments are okay to be outside of your functions.
- **Read the entire document before starting this assignment.**

The goal of this homework is for you to enhance your understanding of strings, indexing, and lists. The homework will consist of 5 functions for you to implement. You have been given `HW04.py` skeleton file to fill out. However, below you will find more detailed information to complete your assignment. Read it thoroughly before you begin.

Hidden Test Cases: In an effort to encourage debugging and writing robust code, we will be including hidden test cases on Gradescope for some functions. You will not be able to see the input or output to these cases. Below is an example output from a failed hidden test case:

```
Test failed: False is not true
```

Written by Naomi Eskinder Woudneh (nwoudneh3@gatech.edu) & Kathleen Butler (kbutler36@gatech.edu)

Fix Tickets

Function Name: fixTickets()
Parameters: ticketNumber (`str`)
Returns: fixedTicket (`str`)
Description: You've won a Six Flags ticket number for this weekend!! Unfortunately, Six Flags messed up the ticket. Each uppercase letter in the ticket number is meant to be lowercase, and each lowercase letter is meant to be uppercase. Write a function that takes in the ticket number (a string of letters and numbers) and returns a string with the corrected ticket number; that is, with each lowercase letter now uppercase, and each uppercase letter now lowercase.

Note: The **.isupper()** and **.islower()** methods might be helpful here.

```
>>> fixTickets("tC2Rhf5Y7b")
'Tc2rHF5y7B'
```

```
>>> fixTickets("r88YuHHn4V")
'R88yUhhN4v'
```

Secret Message

Function Name: secret()
Parameters: message (`str`)
Returns: secret message (`str`)
Description: You and your friends want to send secret messages to each other so you reverse the order of each word within each sentence you write. It's tiresome to do it in your head all the time so write a function that takes in a message as a string and returns a string that contains each word in reverse order.
Hint: The **.split()** method will be useful here.

```
>>> secret("I put in the milk before the cereal")
'I tup ni eht klim erofeb eht laerec'
```

```
>>> secret("I like pineapple on pizza")
'I ekil elppaenip no azzip'
```

Count Tickets

Function Name: countTickets()
Parameters: aList (`list`)
Returns: total (`int`) or error message (`str`)
Description: SCPC has put you in charge of printing the tickets for the six flags event happening this weekend. It's an exciting job but you need to figure out how many tickets to print. Every student is allowed to bring two guests (at most) to the six flags event. This means each student can only purchase a maximum of three tickets. Write a function that takes in a list which contains the name of the ticket buyer and the number of tickets they want to purchase. Return the total number of tickets that need to be printed out.

If anyone in the list ordered more than three tickets, then return a string with an error message that says:

```
"Sorry {name of buyer}, but you can only get a maximum of three tickets per person."
```

If there is more than one person who ordered more than three tickets, then return the error message using the name of the first person who ordered above the maximum.

Note: The list will always follow the order [name, # of tickets, name, # of tickets, ...]

```
>>> countTickets(["Matthew", 1, "Carolyn", 2, "Brian", 2])
5
```

```
>>> countTickets(["Jay", 3, "Parul", 1, "Nikhila", 4, "Josh", 4])
'Sorry Nikhila, but you can only get a maximum of three tickets per person.'
```

Fieldtrip Roster

Function Name: fieldTripRoster()
Parameters: friendList (`list`)
Returns: nameList (`list`)
Description: You are organizing a field trip with your friends and are given a list with sublists. Each sublist includes the name of your friend (`str`), their age (`int`), and how much money they are bringing (`float`). The elements of the sublists are in no particular order, and some friends may be repeated, appearing in multiple sublists. From this jumbled list of lists, you wish to create a list of just your friends' names. Write a function that takes in a list of lists as described, and returns a list of the names of your friends that are coming on the field trip,

with no names repeated, and sorted in alphabetical order.
Hint: The **.sort()** method or **sorted()** function will be useful here.

```
>>> fieldTripRoster([["Kathleen", 26, 40.0], [19, "Naomi", 52.0],
                     ["Kathleen", 26, 40.0], ["Arushi", 22, 36.0],
                     [22, 68.0, "Arvin"], [21, 70.0, "Alexa"],
                     [21, "Craig", 92.0], ["Arushi", 22, 36.0],
                     [21, 92.0, "Craig"]])
['Alexa', 'Arushi', 'Arvin', 'Craig', 'Kathleen', 'Naomi']
```

```
>>> fieldTripRoster([[22, "Shane", 103.0], ["Fareeda", 19.0, 20],
                     ["Elizabeth", 11.0, 65], [15.0, 20, "Alex"],
                     [23, "Elizabeth", 88.0], [72.0, 21, "Fareeda"]])
['Alex', 'Elizabeth', 'Fareeda', 'Shane']
```

Check Sublist

Function Name: isSublist()
Parameters: myList (`list`), otherList (`list`)
Returns: True or False (`bool`)
Description: You and the other SCPC members gathered up the names of people interested in the six flags event so you want to make one big list. However, you think that some of the lists your friends gathered could be sublists of your list. Write a function that checks whether `otherList` is a sublist of `myList`. `otherList` is a sublist of `myList` if all of its elements are in myList in the same consecutive order.

Note: If two lists are equal, they are sublists of one another. Additionally, if a list is empty, it is always a sublist of the other.

```
>>> myList = ["Carolyn", "Anthony" ,"Fareeda" , "Kathleen", "Nelson", "Shane"]
>>> otherList = ["Fareeda", "Kathleen", "Nelson"]
>>> isSublist(myList, otherList)
True
```

```
>>> myList = ["Arvin" , "Arushi" , "Naomi", "Michael", "Nikhila" , "Assata"]
>>> otherList = ["Arushi" , "Naomi" , "Assata" , "Nikhila"]
>>> isSublist(myList, otherList)
False
```

Grading Rubric

Function	Points
fixTickets()	20
secret()	20
countTickets()	20
fieldtripRoster()	20
isSublist()	20
Total	**100**

Provided

The `HW04.py` skeleton file has been provided to you. This is the file you will edit and implement. All instructions for what the functions should do are in this skeleton and this document.

Submission Process

For this homework, we will be using Gradescope for submissions and automatic grading. When you submit your `HW04.py` file to the appropriate assignment on Gradescope, the autograder will run automatically. The grade you see on Gradescope will be the grade you get, unless your grading TA sees signs of you trying to defeat the system in your code. You can re-submit this assignment an unlimited number of times until the deadline; just click the "Resubmit" button at the lower right-hand corner of Gradescope. You do not need to submit your `HW04.py` on Canvas.