Homework 9 - Recursion

CS 1301 - Intro to Computing - Fall 2024

Important

- Due Date: **Thursday, November 14th, 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 Ed Discussion
  - 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 recursion. Recall from class that recursion is when a function calls itself. Using recursion allows us to mimic loops, but you'll see in this homework that recursion may even be more helpful.

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
```

To our mentor, Head TA, and, most of all, our friend, this one's for you, with all our love.
- V&P

Written by: Vaidehi Rathod & Priya Annapureddy

Chris' Home Team Advantage

Function Name: eaglesVsJags()
Parameters: scores (`list`)
Returns: phillyPoints (`int`)
Description: As a big football fan, Chris will never miss a Philadelphia Eagles game! Ahead of the matchup aganist the Jacksonville Jaguars this week, he wants to calculate how many points the Eagles might win.

Write a function that takes in a list of tuples with scores from past Eagles v. Jaguars games in the format `[(team, score), ...]` and **returns** the total number of points the Eagles scored.

Note: `"E"` represents the Eagles and `"J"` represents the Jaguars.

```
>>> eaglesVsJags([("E", 7), ("E", 3), ("J", 3), ("E", 7), ("E", 3)])
20
```

```
>>> eaglesVsJags([("J", 3), ("J", 3), ("E", 7), ("E", 7), ("J", 7)])
14
```

Chris Goes Worldwide

Function Name: nextDestination()
Parameters: hiddenDestination (`str`)
Returns: wheresNext (`str`)
Description: Chris spent a unforgettable summer abroad in Singapore. Since he's graduating soon, he's looking for another international experience.

To figure out where Chris is going next, write a function that takes in a string containing different types of characters and **return** a clean string containing **only letters**.

```
>>> nextDestination("2J-0a-2p-4a-n!")
"Japan"
```

```
>>> nextDestination("S!p!a2023i!n!")
"Spain"
```

Chris' Scooter Diaries

Function Name: canCrash()
Parameters: peopleAndThings (`list`)
Returns: notSafe (`list`)
Description: Tired of making the walk from his apartment to the College of Computing, Chris now flies to and fro with his new scooter. But, he wasn't always so scooter-savvy...

Write a function that takes in a list of people and things and **returns** a sorted list of items from it that he could crash into. Chris is only prone to crash into the items that have an odd amount of letters in their name.

```
>>> canCrash(['Pedestrians', 'Car', 'GT Stinger', 'Cone'])
['Car', 'Pedestrians']
```

```
>>> canCrash(['Josh', 'Arman', 'Audrey', 'Andrew'])
['Arman']
```

East Coast Chris

Function Name: travelLog()
Parameters: cityList (`list`)
Returns: visitedDict (`dict`)
Description: Chris has been up and down the East Coast more times than he can count. Help him create a travel log of all the places he's visited.

Given a list of tuples containing information about different East Coast cities in the format `[(state, city, isVisited), ...]`, write a function that **returns** a dictionary mapping states to a list of their cities, only if Chris has visited them before.

Note: Make sure each list of cities in your dictionary are sorted.

```
>>> travelLog([('ME', 'Portland', False), ('MA', 'Boston', True),
               ('MD', 'Baltimore', True), ('NY', 'New York', True)])
{'MA': ['Boston'], 'MD': ['Baltimore'], 'NY': ['New York']}
```

```
>>> travelLog([('FL', 'Tampa', True), ('FL', 'Miami', True),
               ('NC', 'Raleigh', False), ('NC', 'Charlotte', True),
               ('TN', 'Chattanooga', False)])
{FL: ['Miami', 'Tampa'], NC: ['Charlotte']}
```

Cafe Chris

Function Name: bobaBuilder()
Parameters: menuItems (`list`)
Returns: finalItem (`str`)
Description: Chris is opening his very own boba cafe in Tech Square: Cafe Chris! The menu has a special CS1301 twist, of course.

Given a menu list which contains both strings and sublists, write a function that combines the strings from the menu list in order to find out Chris' speciality tea creation. Then, **return** the combined string.

Hint: How can you configure your recursive step to deal with the sublists?

```
>>> bobaBuilder(["Mango ", "Green ", "Tea"])
"Mango Green Tea"
```

```
>>> bobaBuilder(["Milk ", ["Tea "], ["with ", "Pearls"]])
"Milk Tea with Pearls"
```

Grading Rubric

Function	Points
eaglesVsJags()	20
nextDestination()	20
canCrash()	20
travelLog()	20
bobaBuilder()	20
Total	**100**

Provided

The `HW09.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 `HW09.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 `HW09.py` on Canvas.