Homework 9 - Recursion

CS 1301 - Intro to Computing - Fall 2022

Important

- Due Date: **Tuesday, November 8th, 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 recursion. Recall from class that recursion is when a function calls itself. Using recursion allows us to mimic using for loops (and in some cases, can even be more useful). The following homework requires the use of recursion for every problem. **NOTE: Using for/while loops instead of recursion for any of the homework problems will result in a 0 for that problem.**

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 Paige Holland (pholland7@gatech.edu) & Ramya Subramaniam (ramyasub@gatech.edu)

Weekly Timesheet

Function Name: hoursWorked()
Parameters: clockedHours (`list`)
Returns: totalHours (`int`)
Description: You work for the College of Computing and want to calculate how many hours you worked during the week. Write a function that takes in a list (`list`) of times (`int`), and returns the total hours (`int`) you worked during the week. Assume all times in the list are in the unit of hours and all times are positive.

```
>>> hoursWorked([2, 4, 1, 4, 6])
17
```

```
>>> hoursWorked([1, 3, 6, 12])
22
```

Secret Location

Function Name: secretLocation()
Parameters: location (`str`)
Returns: decodedLocation (`str`)
Description: Your friends are having a secret meeting to discuss throwing a suprise birthday party! Write a function that takes in a secret location (`str`) and returns the decoded location (`str`). In order to decode the secret meeting location, you need to **recursively** remove all characters that are not lowercase letters or spaces. If there are no lowercase letters or spaces in the secret location, then you should return the empty string.

```
>>> secretLocation("!cSKu1lL3c$")
'culc'
```

```
>>> secretLocation("67v@aLLn## 0l3e@Rar")
'van lear'
```

Spring Registration

Function Name: springRegistration()
Parameters: originalCRNs (`list`)
Returns: finalCRNs (`list`)
Description: You are preparing which classes you want to register for next semester by looking up their CRNs. To ensure a smooth registration process during your time ticket, you want to make sure that you are not accidentally entering the same CRN more than once. Write a recursive function that takes in a list of CRNs (`list`) and returns a list of CRNs without any duplicates (`list`). If the list passed in is empty, then return the empty list.

Hint: Start by checking the last element first in order to get the correct order.

```
>>> springRegistration([22275, 25594, 21942, 22275])
[25594, 21942, 22275]
```

```
>>> springRegistration([21942, 25594, 22907, 22275])
[21942, 25594, 22907, 22275]
```

Ponce City Plans

Function Name: poncePlanner()
Parameters: restaurantChoices (`list`)
Returns: taFavorites (`dict`)
Description: The TAs are going to Ponce City Market! Each TA wants to go to their favorite restaurant. Write a function that takes in a list of tuple pairs (`list`), with each tuple holding a TAs name (`str`) and their favorite restaurant (`str`), that recursively creates and returns a dictionary (`dict`) with each TAs name mapped to the restaurant they want to go to. If no tuples are in the passed in list, an empty dictionary should be returned. You may assume that each TA will not be in the list more than once.

```
>>> restaurantChoice = [("Paige", "Dancing Goats"), ("Fareeda", "Botiwala"),
                    ("Ramya", "Minero"), ("Jane", "Pancake Social")]
>>> poncePlanner(restaurantChoice)
{'Jane': 'Pancake Social',
 'Ramya': 'Minero',
 'Fareeda': 'Botiwala',
 'Paige': 'Dancing Goats'}
```

```
>>> restaurantChoice = [("Craig", "Pizza Jeans"), ("Aryan", "King of Pops"),
                        ("Josh", "Vietvana")]
>>> poncePlanner(restaurantChoice)
{'Josh': 'Vietvana',
 'Aryan': 'King of Pops',
 'Craig': 'Pizza Jeans'}
```

Draw Rectangle

Function Name: drawRectangle()
Parameters: width (`int`), height (`int`)
Returns: None (`NoneType`)
Description: You are interested in how you can utilize programming to create visual art. Write a function that takes in a width (`int`) and height (`int`) and **recursively** draws the *outline* by printing out an asterisk(`*`) or space for each unit of the rectangle. If the width passed in is less than 3, then you should print the error message `"You're cutting corners!"`. You may assume that the width and heights passed in will **always be equal**.

Hint: Each row except the first or last will contain `(width - 2)` spaces surrounded by an asterik on both sides.

```
>>> drawRectangle(4, 4)
****
*  *
*  *
****
```

```
>>> drawRectangle(2, 2)
You're cutting corners!
```

Grading Rubric

Function	Points
hoursWorked()	20
secretLocation()	20
springRegistration()	20
poncePlanner()	20
drawRectangle()	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.