Homework 6 - Dictionaries

CS 1301 - Intro to Computing - Fall 2024

Important

- Due Date: **Thursday, October 10th, 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 to introduce you to the wonderful, endless world of dictionaries. The homework will consist of 5 functions for you to implement. You have been given the HW06.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: Shruti Marx, Saleh Alsedrah, & Priya Annapureddy

National Taco Day

Function Name: tacoBuilder()
Parameters: shoppingList (`list`), inventory (`list`), tacoIngredients (`list`)
Returns: tacoDict (`dict`)
Description: October 4th is National Taco Day! You're throwing a taco fiesta to celebrate.

Write a function that takes in three lists: a shopping list of of tuples you want to buy in the format `[(item, quantity)...]` , an inventory list of ingredients you already have in the format `[(item, quantity)...]` , and a list of taco ingredients you want to use for your party. **Return** a single taco dictionary that maps ingredients to their quantity *after* shopping, only if they are in the taco ingredients list. If there are ingredients in the taco ingredient list that you won't have after shopping, their value in the final dictionary should be 0.

Note: Remember that because dictionaries are unordered, the order of your dictionary may differ from the test cases below.

```
>>> shoppingList = [('tortilla', 10), ('beans', 3), ('beef', 9), ('ice cream', 7)]
>>> inventory = [('tortilla', 3), ('cheese', 7), ('cereal', 0)]
>>> tacoIngredients = ['tortilla', 'cheese', 'beans', 'beef']
>>> tacoBuilder(shoppingList, inventory, tacoIngredients)
{'tortilla': 13, "cheese": 7, "beans": 3, "beef": 9}
```

```
>>> shoppingList = [('taco shells', 20), ('lettuce', 2),
                        ('apples', 10), ('salsa', 3)]
>>> inventory = [('lettuce', 2), ('salsa', 1), ('apples', 7)]
>>> tacoIngredients = ['taco shells', 'lettuce', 'sour cream', 'salsa']
>>> tacoBuilder(shoppingList, inventory, tacoIngredients)
{'taco shells': 20, "lettuce": 4, "sour cream": 0, "salsa": 4}
```

National Do Something Nice Day

Function Name: goodDeeds()
Parameters: deeds (`dict`)
Returns: numDeeds (`dict`)
Description: October 5th is National Do Something Nice Day! You take your national holidays seriously, so for some days of the week, you vow to do a good deed!

Write a function that takes in a dictionary that maps days of the week to a corresponding list of tuples. Each tuple represents a specific good deed with its how many times its done in the format `(deed, quantity)`. **Print** out the day of the week with the most good deeds done, and then **return** a dictionary with the good deed mapped to the number of times that deed was performed during the week.

```
>>> goodDeeds({"Monday":[("Doing Dishes", 2), ("Buy Flowers", 5)],
              "Tuesday":[("Call Friends", 33),("Doing Dishes", 21)],
              "Wednesday":[("Write a Letter", 32), ("Buy Flowers", 21)]})
Tuesday
{'Doing Dishes': 23, 'Buy Flowers': 26, 'Call Friends': 33, 'Write a Letter': 32}
```

```
>>> goodDeeds({"Thursday":[("Doing Dishes", 9), ("Buy Flowers", 3)],
              "Friday":[("Call Friends", 12),("Doing Dishes", 21)],
              "Saturday":[("Write a Letter", 32), ("Buy Flowers", 30)],
              "Sunday":[("Wash Car", 4), ("Call Friends", 32)]})

Saturday
{'Doing Dishes': 30, 'Buy Flowers': 33, 'Call Friends': 44,
'Write a Letter': 32, 'Wash Car': 4}
```

National LED Light Day

Function Name: equivalentRatios()
Parameters: efficiency (`int`), bulbCandidates (`dict`)
Returns: betterBulbs (`dict`)
Description: October 7th is national LED light day! Luckily, you're in a home improvement mood. You decide to replace all of your current LED lights with ones that are more efficient.

Write a function that takes in two parameters: your current light bulb efficiency score and a dictionary mapping some of the best light bulbs currently on the market to a tuple of their details, formated as `{bulbName:(actualWattage, incandescentEquivalent), ...}`. You must calculate the efficiency of each bulb in the dictionary by dividing their incandescent equivalent by their actual wattage. If a bulb has a higher efficiency than you currently have, it must be added to a dictionary in the format `{bulbName: efficiency}`. **Return** the dictionary of better bulbs, if you find any. Otherwise, **return** the string `"I must've upgraded them last international LED light day!"`

Note: The efficiencies must be rounded to 2 decimal places.

```
>>> bulbCandidates = {"Philips Hue Smart Bulb":(10,70),
                      "Philips Dimmable LED A19 Bulb":(5,40),
                      "GE Reveal LED Bulb":(11,75), "Cree LED A19 ":(9.5,60)}
>>> equivalentRatios(6.5, bulbCandidates)
{'Philips Dimmable LED A19 Bulb': 8.0,
'Philips Hue Smart Bulb': 7.0, 'GE Reveal LED Bulb': 6.82}
```

```
>>> bulbCandidates = {"EcoSmart LED Daylight Bulb":(8,60),
                      "Govee Smart Light Bulb":(18,150),
                      "Sphoon G40 LED Bulb":(1.5,15)}
>>> equivalentRatios(8.5, bulbCandidates)
{'Sphoon G40 LED Bulb': 10.0}
```

National Pierogi Day

Function Name: makePierogi()
Parameters: pierogiTypes (`dict`)
Returns: pierogiIngredients (`dict`)
Description: October 8th is National Pierogi Day! Pierogi is a sweet or savory dish similar to boiled dumplings, and you're going to try making some.

Write a function that takes in a dictionary mapping a type of pierogi to it's ingredients, and **return** a dictionary of ingredients mapped to a **sorted** list of what pierogis they're included in.

```
>>> makePierogi({"Vegetarian": ("Carrot", "Celery", "Spinach"),
                 "Meat": ("Chicken", "Carrot")})
{'Carrot': ['Meat', 'Vegetarian'], 'Celery': ['Vegetarian'],
'Spinach': ['Vegetarian'], 'Chicken': ['Meat']}
```

```
>>> makePierogi({"Sweet": ("Strawberry", "Chocolate"),
                 "Fruit": ("Strawberry", "Peach")})
{'Strawberry': ['Fruit', 'Sweet'], 'Chocolate': ['Sweet'], 'Peach': ['Fruit']}
```

National Cake Decorating Day

Function Name: canMake()
Parameters: recipes (`dict`), ingredients (`dict`)
Returns: canMakeCake (`bool`)
Description: October 10th is National Cake Decorating Day! As the owner of a bakery, you are preparing to host a cake decorating class.

Write a function that takes in two parameters: a dictionary mapping recipes to a list of ingredients needed to make, formatted as `{recipe: [(ingredient, quantity), ...]}`, and a dictionary of ingredients you have, formatted as `{ingredient: quantity, ...}` . If you have enough ingredients to make *every* cake topping in the recipes dictionary, **return** True. Otherwise, **return** False.

```
>>> recipes = {"Butter Cream": [("Butter", 2), ("Powdered Sugar", 3)],
                "Caramel": [("Sugar", 2), ("Butter", 2)]}
>>> ingredients = {"Butter": 5, "Powdered Sugar": 4, "Sugar": 7}
>>> canMake(recipes, ingredients)
True
```

```
>>> recipes = {"Strawberry Jam": [("Strawberries", 100), ("Sugar", 2)],
                "Hot Fudge": [("Chocolate", 7)]}
>>> ingredients = {"Strawberries": 200, "Chocolate": 20}
>>> canMake(recipes, ingredients)
False
```

Grading Rubric

Function	Points
tacoBuilder()	20
goodDeeds()	20
equivalentRatios()	20
makePierogi()	20
canMake()	20
Total	**100**

Provided

The `HW06.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 `HW06.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 `HW06.py` on Canvas.