Homework 11

CS 1301 - Intro to Computing - Fall 2022

Important

- Due Date: **Tuesday, December 6th, 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.**

For this assignment, you will be building classes based on Thanksgiving. Thanksgiving is national holiday celebrated on the fourth Thursday of November each year in the United States where people across the country give thanks for their family, friends, and more. There will be three classes working together: a People class, a Food class, and an Activities class.

Each of these classes will have attributes and methods, as described below. You have been provided with a file that has the beginning of these classes. You are responsible for filling in the rest, and the methods you need will be clearly listed out in the grading rubric. **Please read the entire assignment before writing your code in order to understand how all of the classes interact with each other!**

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 Cynthia Medlej (cmedlej3@gatech.edu), Jane Crowley (jcrowley38@gatech.edu)

The Gift Map

Function Name: santaHelper()
Parameters: kids (`list`)
Returns: destinations (`dict`)
Description: Santa is overwhelmed this year and needs your help. You will be given a list of tuples in the following format (name, isNice, location): name is the kid's name (`str`), isNice is a boolean (`bool`) - True if the kid is nice and False if the kid is naughty -, and location the kid's address (`str`).
Write a function that will **recursively** return a dictionary mapping each location to a **sorted** list of the kids who live there only if they are nice.

```
>>> kids = [
        ("Ralph", True, "Houston"),
        ("Bailey", False, "New York"),
        ("Bill", True, "San Francisco")]

>>> santaHelper(kids)
{"Houston" : ["Ralph"], "San Francisco" : ["Bill"]}
```

```
>>> kids = [
        ("Joe", True, "New York"),
        ("Aiden", True, "Atlanta"),
        ("Emma", True, "New York")]


>>> santaHelper(kids)
{'New York': ['Emma', 'Joe'], 'Atlanta': ['Aiden']}
```

Santa's got mail

Function Name: myWishlist()
Parameters: stores (`dict`)
Returns: None
Description: You learned Python this semester and you feel like writing your wishlist for santa with a pen and paper is way too outdated. Given a dictionary mapping each store to the items they sell and their prices, write a function that outputs a file named `"wishlist.txt"` that follows the format below. Each store should be ranked in ascending order based on the price of the item it sells.Make sure there is **no extra lines** at the end of your file.

`wishlist.txt:`

```
My WishList

Item#1:

Store#1
Store#2
...

Item#2:

Store#1
Store#2
...
```

Note: You may assume that no two items will have the same price.
Hint: You can sort tuples by their first item!

```
>>> stores = {"Nordstrom": [("Jacket",53),("Hat",61)],
          "Bloomingdales":[("Jacket",46),("Hat",89),("Shoes",77)]}
>>> myWishList(stores)
```

Output of `wishlist.txt`:

```
My WishList

Jacket:

Bloomingdales
Nordstrom
```

```
Hat:

Nordstrom
Bloomingdales

Shoes:

Bloomingdales
```

```
>>> stores = {"Apple": [("MacBook",2000),("iPhone14",1500)],
          "Barnes and Nobles":[("GTMerch",300000000),("MacBook",2200)]}
>>> myWishList(stores)
```

Output of `wishlist.txt`:

```
My WishList

MacBook:

Apple
Barnes and Nobles

iPhone14:

Apple

GTMerch:

Barnes and Nobles
```

Hidden Gift

Function Name: getMessage()
Parameters: presaleCode (`int`)
Returns: message (`str`)
Description: Taylor Swift recently posted on her Instagram story saying that the ticket presale code you receive has a hidden message in it. In order to decode it, you have to convert the decimal into hexadecimals. Write a function that takes in a presaleCode(`int`) and returns a message(`str`) by decoding the presaleCode.

```
>>> getMessage(14598366)
>>> 'DEC0DE'
```

```
>>> getMessage(12648430)
>>> 'C0FFEE'
```

Do You Want To Build A Snowman

Function Name: buildFrosty()
Parameters: materials (`dict`)
Returns: snowManStr (`str`)
Description: You and your friends want to build a snowman and are figuring out if you have enought supplies to make one. In order to build a snowman, you need: 2 sticks, 10 rocks, 1 carrot, and 1 hat. Write a function that returns the string, `"We have enough materials to build the snowman :)"` if you have enough materials, or returns `"Not enough materials :("` if you don't have enough materials.

```
>>> materials = {"Ramya": {"sticks": 1, "rocks": 3, "hat": 1},
             "Jane": {"carrot": 1, "scarf": 1, "shoes": 2, "rocks": 8},
             "Anastasiya": {"sticks": 2, "buttons": 3, "carrot": 10}}
>>> buildFrosty(materials)
'We have enough materials to build the snowman :)'
```

```
>>> materials = {"Cynthia": {"sticks": 5, "rocks": 3, "carrot": 1},
             "Harshith": {"carrot": 1, "scarf": 1, "socks": 3, "rocks": 3},
             "Josh": {"rocks": 2, "buttons": 3, "hat": 1}}
>>> buildFrosty(materials)
'Not enough materials :('
```

New Elf In Town

Function Name: guideBuddy()
Parameters: locationDict(`dict`), movementsList(`list`)
Returns: newRoutine (`str`)
Description: Buddy the Elf has arrived in NYC to spread Christmas magic. He's having trouble describing where he is but he has a list of each movement he made. Given that he started at position (0,0) and a dictionary of locations with their coordinate, determine the closest location to him and return a string in the format `"Go to {nearest location}!"`. To determine the distance between two locations, use the provided distance function on the current location of Buddy and the locations given from the dictionary.

Note: Assume that no two locations will be the same distance from Buddy.
Hint: A movement to the left would add (0,-1) to his placement, while a movement up would add (1,0).

```
>>> locationDict = {"Central Park": [3,4], "Empire State Building": [-2,4],
                    "Statue of Liberty": [-1, 5], "Rockefeller Center": [4, 1]}
>>> movementsList = ["right", "up", "right", "right", "down",
                     "down", "down", "left", "up"]
>>> guideBuddy(locationDict, movementsList)
'Go to Empire State Building!'
```

```
>>> locationDict = {"Central Park": [3,4], "Empire State Building": [-2,4],
                    "Statue of Liberty": [-1, 5], "Rockefeller Center": [4, 1]}
>>> movementsList = ["right", "left", "left", "left",
                     "up", "up", "up", "left", "left"]
>>> guideBuddy(locationDict, movementsList)
'Go to Rockefeller Center!'
```

Grading Rubric

Function	Points
santaHelper()	20
myWishlist()	20
getMessage()	20
buildFrosty()	20
guideBuddy()	20
Total	**100**

Provided

The `HW11.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 `HW11.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 `HW11.py` on Canvas.