15-110 Recitation Week 7

Reminders

- 03/12 Tue - Check3/HW3 revisions due (Tuesday after break)
- Reci feedback form
- Have a restful and rejuvenating break!

Overview

- Big-O Exercise
- For Loop Review
- Dictionary Review
- Tree Code Writing
- Dictionary Code Writing

Problems

BIG-O EXERCISE

Calculate the Big-O for the following examples:

Returning the last character in a string

```
def powersOfTwo(n): # n = n
    m = 1
    while m <= n:
        print(m)
        m *= 2
```

```
def foo(L): # len(L) = n
    if L == []:
        return 0
    else:
        L.append(L[0])
        n = L.index(10)
        L.pop(0)
        return n
# .index(), .pop() are O(n) worst
case!
```

```
#You are guaranteed L is a nxn 2D
list
def tripleLoop(L):
    for i in range(20):
        for row in L:
            for elem in row:
                print(elem)
```

FOR EACH LOOP REVIEW

Notes on Loops::

Problem:

Use the following code to answer the questions:

```
s = "15-110"
for i in range(len(s)):
    print(i)
for i in s:
    print(i)
```

What does the code print?

What is the type of i for each loop?

DICTIONARY REVIEW

Notes on dictionaries:

Here is an example of a type of problem that uses dictionaries. Read through the problem statement and solution and note the key points of the code.

Problem:
Kelly's Bakery is doing an inventory of their freshly baked goods. This morning, they baked new items and now they need to update their inventory to represent these items. You are given a dictionary that represents the inventory at Kelly's Bakery, which maps the name of the item to how many items of that baked good are available. Write the function updateInventory(d, newItems) that takes the current inventory and a new dictionary called newItems and updates it accordingly. The function should also handle the case that there is an item in newItems that doesn't exist in d.

Solution:
```
def updateInventory(d, newItems):
  for item in newItems:
     if item in d:
        d[item] += newItems[item]
     else:
        d[item] = newItems[item]
  return 33
```

TREE CODE WRITING

Write the function addEvenLeaves(t) that takes in a dictionary representation of a tree (you can assume it will have at least 1 node) and returns a sum of **only** the even values held by leaves.

```
def addEvenLeaves(tree):
    # base case: leaf node
    if ________________ and ______________:
        # check if leaf’s value is even
        if ______________:
           # returns the leaves value
           return ______________
        else:
           # what should you return if the leaf isn’t even?
           return ____
    else:
        value = 0
        # recursive case if left subtree is not None
        if ____________:
            value += _____________________
        # recursive case if right subtree is not None
        if ____________:
            value += _____________________
        return value
```

DICTIONARY CODE WRITING

Given a dictionary that maps teams like CMU, Pitt, OSU, PennState, and another unspecified number of football teams, to the number of wins and losses they have (represented as [wins, losses]), and an integer representing the minimum amount of games to be considered, we want to return the team with the best win percentage and that has played enough games. There will be no ties. For example,

bestTeam({ "CMU" : [1, 10], "Pitt" : [7, 10], "OSU" : [10, 6], "PennState" : [2, 1] }, 5) returns "OSU"

```
def bestTeam(winsLosses, minGames):

    bestTeam = ____________

    bestRatio = ___________

    for team in winsLosses:

        wins = ___________________

        losses = __________________

        gamesPlayed = _________ + _________


        #check if played enough games

        if _____________ >= minGames:

            winRatio = ____________ / gamesPlayed

            if _____________ > bestRatio:

                bestRatio = ________________

                bestTeam = _________________

    return _____________
```