Name: _____ Recitation: _____ Andrew Id: _____

15-112 Spring 2019 Quiz 7

Up to 20 minutes. No calculators, no notes, no books, no computers. Show your work!

1. (30 points) **Free Response:** Write the function `getHoursLogged(logs)`, which takes `logs`, a list of (time, person) tuples, and returns a dictionary mapping each person in the logs to the total number of minutes they worked. The logs are formatted in the same way as recitation- each tuple either represents a person checking into work, or a person checking out of work. Each person who appears in the list has **exactly two logs**, so the total time a person works is the amount of time that passes between their check-in and check-out. You are guaranteed that the tuples are ordered by the time values.

   Example: if the function is called on the list `[(0, 'Spongebob'), (10, 'Krabs'), (30, 'Squidward'), (55, 'Krabs'), (250, 'Squidward'), (300, 'Spongebob')]` it will return
   `{ 'Spongebob': 300, 'Krabs': 45, 'Squidward' : 220 }`, because Spongebob works from 0 to 300 (300 minutes), Krabs works from 10 to 55 (45 minutes) and Squidward works from 30 to 250 (220 minutes).

   To get full credit, your solution must run in O(N) time, where N = `len(logs)`.

2. (20 points) **Short Answer:** For both of the two functions shown below, write next to each line of the function either the Big-O runtime of the line or the number of times the line loops. Then write the total Big-O runtime of the function in terms of N in the box to the right of the code. You must write in the Big-Os of each line to get full credit.

```
1: def f1(n): # n is an int, N = n                    # Big-O
2:     tmp = n                                       #______
3:     x = 0                                         #______
4:     while tmp > 0:                                #______
5:         lst = list(range(n ** 2))                 #______
6:         lst.reverse()                             #______
7:         tmp = tmp // 10                           #______
```

```
1: def f2(lst1, lst2): # len(lst1) = len(lst2) = N    # Big-O
2:     result = 0                                    #______
3:     for i in range(len(lst1)):                    #______
4:         for j in range(i+1, len(lst1)):           #______
5:             if sum(lst1[i:j]) in lst2:            #______
6:                 result = max(result, sum(lst1[i:j])) #______
7:     return result                                 #______
```

Built-in Big-O Runtimes	
General	
`len(item)`	$O(1)$
`item[i]`	$O(1)$
List of length N	
`lst[i:j:k]`	$O((j-i)/k)$
`sum(lst)`	$O(N)$
`min(lst) / max(lst)`	$O(N)$
`lst.reverse()`	$O(N)$
`item in lst`	$O(N)$

3. (15 points) **Short Answer:** The following three questions have to do with hash functions and sets. Choose only **one** answer for each.

   (i) What is a hash function?
   - ◯ A function that turns a mutable data type into an immutable data type
   - ◯ A function that turns an immutable value into an integer
   - ◯ A function that finds a value in a list in constant time

   (ii) How is a hashed item put into a set?
   - ◯ The hash value is used to create an index into a list; the item is placed in an inner list at that index
   - ◯ The hash value is placed into the set instead of the item
   - ◯ The hash value is stored as a key and the item is stored as a value, just like in a dictionary

   (iii) How is it possible to see if an item is in a set in O(1) time?
   - ◯ Since a set has a constant size, we can scan the entire set for the hash value in O(1) time
   - ◯ Hashing the item will tell us which index to look into, and we can search the inner list in O(1) time
   - ◯ We can't see if an item is in a set in O(1) time

4. (15 points) **Short Answer:** The following program returns the number of repeated elements in lst, but it's less efficient than it could be. Briefly describe how to change the program to make it more efficient without completely changing the algorithm. Your answer must change the function family of the program's runtime.

```
def countRepeats(lst):
    count = 0
    uniqueItems = []
    for item in lst:
        if item not in set(uniqueItems):
            uniqueItems.append(item)
        else:
            count += 1
    return count
```

5. (20 points) **Code Tracing:** Indicate what the following program prints. Place your answer (and nothing else) in the box under the code.

```
import copy
def ct(a):
    b = copy.copy(a)
    c = copy.deepcopy(a)

    b[0][1] = 4
    a.insert(0, "OOP")
    c[1].pop()
    a[-1] = "quiz"
    b.pop(1)
    c.extend([1, 2])

    print("a", a)
    print("b", b)
    print("c", c)

ct([ [ 3, 5 ], [ "set", "go" ], [ True ] ])
```