Online 1. _______/20
Online 2. _______/15
Online 3. _______/30
Online 4. _______/35

Total _______/100

ANSWERS
Athena User Name

ANSWERS________________________
Full Name

Recitation hour

The quiz for 6.0001 is on MITx. **Do not write your answers on this paper.** Only fill out this front page with your name, username, and recitation.

You are allowed 1 page (2 sides total) on which you can write anything. On your computer, you are only allowed to have the MITx website open and Anaconda/IDLE. Close everything except for these. **If the test proctors see that you have anything else open on the computer or if we believe you are trying to cheat, you will be asked to leave immediately and given a zero on the exam.**

**You have 90 minutes to complete the exam.**

You must check out with an exam proctor after you have finished your exam but before you leave. **Hand in this paper cover sheet when you checkout.** Any student who does not check do this will receive a 0 on the exam.
Problem 1

1. Suppose x = "pi" and y = "pie". The line of code x, y = y, x will swap the values of x and y, resulting in x = "pie" and y = "pi".

2. Suppose x is an integer in the following code:
   ```python
def f(x):
    while x > 3:
        f(x+1)
```

   For any value of x, all calls to f are guaranteed to never terminate.

3. A Python program always executes every line of code written at least once.

4. Applying bisection search as opposed to linear search usually significantly increases the speed of a program for large inputs.

5. Suppose you have two different functions that each assign a variable called x. Modifying x in one function means you always modify x in the other function for any x.

6. Assume a break statement is executed inside a function. The function call will always terminate without executing any remaining code inside that function.

7. Let d be a dictionary. If d[a] == d[b] then a == b.

8. Suppose you run the following code:
   ```python
def f(L):
    L.append(4)
    x = [1,2,3]
    f(x)
```

   The variable x is an alias for the variable L.

9. Assume the statement `s[1024] = 3` does not produce an error message. This implies:

   - type(s) can be str
   - type(s) can be tuple
   - type(s) can be list
   - All of the above

10. Consider the following function.
    ```python
def f(x):
    a = []
    while x > 0:
        a.append(x)
        f(x-1)
```

    A new object of type list is created for each recursive invocation of f.
Problem 2

Write a Python function that computes the square root of a float sq within an error of bound using the procedure outlined below. Start with a guess guess = 1.0, and at each step, update the guess using the formula 0.5 * (guess + sq/guess). Code that calculates the square root using any other method will not receive credit. This function takes two floats and returns a single float.

Answer:
```
def iterSqrt(sq, bound):
    '''
    sq: the number to find the square root of (float)
    bound: precision for square root calculation (float)
    '''
    guess = 1.0
    while abs(sq - guess*guess) >= bound:
        guess = 0.5 * (guess + sq/guess)
    return guess
```

Problem 3

You are given a dictionary d that maps integer keys to integer values. Write a Python function that returns a new dictionary that maps integer keys to list values, where:
- the keys are the unique values of d and
- the values are sorted lists, with each list containing all the keys of d that mapped to the value in d.

If d is empty, return an empty dictionary. Your code should not modify d. For example:
If d = {1:3, 5:4, 4:4} then the function should return {3:[1], 4:[4,5]}

Answer:
```
def flipDict(d):
    '''
    d: a dictionary int:int
    returns: a new dictionary int:list where the keys are values of d and
             values are the sorted list of all keys of d with that value
    '''
    newd = {}
    for k in d.keys():
        if d[k] in newd:
            newd[d[k]].append(k)
        else:
            newd[d[k]] = [k]
    for k in newd.keys():
        newd[k].sort()
    return newd
```

Problem 4

You are given two lists, L1 and L2 whose elements are tuples. The tuples are made up of pairs of integers. For example, L1 = [(1,2), (4,1), (6,0)] and L2 = [(3,2), (1,0), (3,3), (2,4)]. Now, you are interested in finding the sum of the elements of each tuple. In L1, the sums would be 3, 5, and 6. In L2, the sums would be 5, 1, 6, and 6. Knowing these sums, you want to find out if the set of sums of L1 are permutations of the set of sums of L2.
Your task is to write a Python function that determines whether the two sets of sums as described above are permutations of each other. In this question, the sums in L1 are a permutation of the sums in L2 when:

- a sum e is in L1 if and only if e is in L2 and
- a sum e is in L2 if and only if e is in L1

i.e. In general, a permutation means that two lists share the exact same elements but not necessarily in the same order (with duplicates being allowed but not counted), or both lists are empty.

Your function will take in two lists, L1 and L2. Assume these lists:

- are not necessarily the same length
- can contain the same tuples more than once
- can be empty

Your function will return True if the sums of the tuples of these lists are permutations of each other or if both lists are empty, and return False otherwise. Your function should only use object types taught so far.

Example 1
L1 = [(1,2),(1,3)]
L2 = [(1,3),(1,2)]

Example 2
L1 = [(1,2),(1,3)]
L2 = [(1,2),(1,3),(1,2)]

Example 3
L1 = [(1,2),(0,3)]
L2 = [(1,2)]

Example 4
L1 = [(1,2),(1,3)]
L2 = [(1,1),(2,0),(3,0)]

Answer:

```python
def isSumPermutation(L1, L2):
    """
    L1 and L2 are lists of tuples, where tuples are pairs of ints
    Returns true if and only if L1 and L2 contain permutations
    of the tuple sums
    """
    sum_L1 = []
    sum_L2 = []
    for i in L1:
        sum_L1.append(i[0]+i[1])
    for i in L2:
        sum_L2.append(i[0]+i[1])
    for i in sum_L1:
        if i not in sum_L2:
            return False
    for j in sum_L2:
        if j not in sum_L1:
            return False
    return True
```