Exam 2 Sample Questions — CS1210, Computer Science I

1. Consider the following functions. (Note: you do *not* need to know anything about function `foo` to answer the questions below.)

```
def q1ab(n):
    result = 0
    i = n
    for i in range(n):
        for j in range(n):
            result = result + (i * foo(j) * foo(i+j))
    return result
```

   a. If function `q1ab(10000)` requires 2 seconds to execute, approximately how long will `q1ab(20000)` take?

   b. Give a tight Big-O running time bound for q1ab. I.e. q1ab(n) = O(?)

```
def q1cd(n):
    result = 1
    i = 1
    while i < n:
        result = result + (2 * i)
        i = i + 1
    for num in range(n):
        result = result + (num - 10)
    return result
```

   c. If function `q1cd(10000)` requires 2 seconds to execute, approximately how long will `q1cd(20000)` take?

   d. Give a tight Big-O running time bound for q1cd. I.e. q1cd(n) = O(?)

2. a. Draw the directed graph corresponding to this adjacency list representation:

```
{"ORD" : ["JFK", "LAX", "SFO"],
 "CID" : ["DEN", "ORD"],
 "DEN" : ["CID", "SFO"],
 "JFK" : ["LAX"],
 "LAX" : ["ORD"],
 "SFO" : []}
```

b. Each of the vertex names represents an airport (e.g. ORD is Chicago-O'Hare). And each edge represents an available direct flight from one airport to another. Complete function printAllDirectFlights(flightGraph) that takes as input a dictionary like in part a and prints lines of the form

`"There is a direct flight from ... to ..."` (where the ...'s are replaced airports),

one for each direct flight represented by the graph.

```
def printAllDirectFlights(flightGraph):
```

3.
```
def selectionSort(L):
    i = 0
    while i < len(L):
        minIndex = findMinIndex(L, i)
        L[i], L[minIndex] = L[minIndex], L[i]
        i = i + 1
        print(L)


def findMinIndex(L, startIndex):
    minIndex = startIndex
    currIndex = minIndex + 1
    while currIndex < len(L):
        if L[currIndex] < L[minIndex]:
            minIndex = currIndex
        currIndex = currIndex + 1
    return minIndex
```

The code above implements selection sort. It is the same code discussed in class and the textbook except that comments have been removed.

Show the printed output resulting from executing:

```
selectionSort([3, 99, -2, 17, 5])
```

4. Complete the following class definition, Box, for representing cardboard boxes with given dimensions, width-by-height-by-depth, that can hold a number of objects weighing, in total, up to the specified maxWeight. Write in the code for the two incomplete methods: addObject and isLargerThan

```
class Box:
    def __init__(self, width, height, depth, maxWeight):
        self.width = width
        self.height = height
        self.depth = depth
        self.maxWeight = maxWeight

        self.objectsInside = []
        self.currentWeightInside = 0
        self.currentNumberOfObjectsInside = 0

    def volume(self):
        return self.width * self.height * self.depth

    def numberOfObjectsInside(self):
        return self.currentNumberOfObjectsInside

    def remainingWeightCapacity(self):
        return self.maxWeight - self.currentWeightInside

    # If the box's weight capacity will not be exceeded,
    #  add given object to the box, updating objectsInside,
    #     currentWeightInside and currentNumberOfObjectsInside
    # Otherwise, print an appropriate message.
    #
    def addObject(self, object, objectWeight):
        ???









    # Return True if the size (by volume) of self is larger than that of the other box
    # Return False otherwise.
    #
    def isLargerThan(self, otherBox):
        ???
```

5. Consider function flipCoins. It's a template for you to refer to - it just "flips coins" but doesn't keep track of or return any values.

```
def flipCoins(numFlips):
    flipNum = 0
    while flipNum < numFlips:
        # consider value of 1 to be heads, 2 to be tails
        value = random.randint(1,2)
        flipNum = flipNum + 1
    return None
```

a. Complete function longestHeadsRun(numFlips) that does numFlips flips and returns the length of the longest "run" of consecutive heads generated during the sequence of flips. NOTE: you should *not* call flipCoins - it's just there to help you remember how to flip coins. Put all the necessary code in your new function.

```
def longestHeadsRun(numFlips):
```

b. What is the Big-O running time bound for longestHeadsRun?

6. a. Consider the problem of finding the *third smallest* element of a given list of numbers. For example, the third smallest in `[1, 99, 7, -3, 3, 10, 12]` is 3.

In lecture and homework, you have seen and used the code below (except for the name "thirdSmallest" used for the second function). Modify/finish the indicated lines of thirdSmallest so that it correctly returns the third smallest element. (IMPORTANT NOTE: you may not call the Python sort/sorted functions.)
The modified algorithm must also have a running time bound better than $O(n^2)$.

```
# return index of min item in L[startIndex:]
# assumes startIndex < len(L)
#
def findMinIndex(L, startIndex):
    minIndex = startIndex
    currIndex = minIndex + 1
    while currIndex < len(L):
        if L[currIndex] < L[minIndex]:
            minIndex = currIndex
        currIndex = currIndex + 1
    return minIndex

def thirdSmallest(L):
    i = 0
    while i < len(L):                                    <--- MODIFY THIS LINE
        minIndex = findMinIndex(L, i)
        L[i], L[minIndex] = L[minIndex], L[i]
        i = i + 1
    return ?                                              <--- FINISH THIS LINE
```

b. What is the Big-O running time bound of thirdSmallest?