Final Exam

22C:2110/3110 Programming for Informatics
Friday, Dec. 19, 2014

45 points total

1. (9 points) Consider the functions:

```
def p1a(n):
    result = 0
    if ((n % 2) == 0) or ((n % 5) == 0):
        return result+1
    return result

def p1b(n):
    result = 0
    if ((n % 2) == 0):
        result = result+1
    if ((n % 5) == 0):
        result = result+1
    return result

def p1c(n):
    if (n%2) == 0:
        return 1
    if (n%5) == 0:
        return 1

def p1d(n):
    result = 0
    if (n%2) == 0:
        result = result + 1
    elif (n%5) == 0:
        result = result + 1
    return result
```

For each item below, say whether or not the specified function returns the same value as p1a on all integer inputs. If you answer "no" give, provide an example input/output values that demonstrate the difference.

p1b:

p1c:

p1d:

2. (9 points) Consider the following functions:

```
def p2a(n):
    result = 0
    for i in range(n+1):
        result = result + i
    return result

def p2b(n):
    result = n
    i = n-1
    while (i >= 0):
        result = result + i
        i = i - 1
    return result

def p2c(n):
    if n == 0:
        return 0
    else:
        return p2c(n-1) + n
```

a. Clearly state, in one or two sentences, what p2a(n) computes, for any given integer n

b. Is p2b(n) == p2a(n) for every n? Carefully explain why or why not.

c. Is p2c(n) == p2a(n) for every n? Carefully explain why or why not.

3. (9 points) We will call a list *unimodal* if it consists of a strictly decreasing sequence of numbers followed by a strictly increasing sequence. *Note: this is a different definition of unimodal than in the version discussed on the last day of class.*) Thus, any unimodal list will have at least three elements and no two consective elements will have the same value.

   Complete function unimodalMin below so that, when given a unimodal list as input, it returns a two-element tuple containing the minimum and the index of the minimum.

   For example, unimodalMax([102, 95, 94, -1, -4, 3]) should return (-4, 4).

```
def unimodalMin(L):
    done = False
    lo = 0
    hi = len(L) - 1
    while (done == False):
        midIndex = (lo + hi) // 2
        midItem = L[midIndex]

        if ________________________________ and ________________________________ :

            done = True

        elif L[midIndex-1] > midItem:

           ________________________________

        else:

           ________________________________


    return ________________________________
```

4. (9 points) Complete the following class definition by adding code for the four methods: getAdjustedHomeworkTotal, getAdjustedHomeworkAverage, getAdjustedTotalPoints, and getAdjustedTotalPerentage

```
class StudentInfo:
    def __init__(self, name, studentID):
        self.name = name
        self.studentID = studentID
        self.homeworkScores = []
        self.examScores = []

    def addHomeworkScore(self, score):
        self.homeworkScores.append(score)

    def addExamScore(self, score):
        self.examScores.append(score)

    # Return the sum of the homework scores, excluding the lowest score.
    # Assume there are always at least two homework scores.
    # (Note: if there are two or more equally lowest scores, only drop one of them.)
    #
    def getAdjustedHomeworkTotal(self):









    # Return a (float) number that's the average of the scores used in the adjusted homework total
    #
    def getAdjustedHomeworkAverage(self):









    # Return the total of all exam scores
    def getExamTotal(self):
        return sum(self.examScores)
```

```
    # Return the total points scored (not including possible points for one dropped homework)
    #
    def getAdjustedTotalPoints(self):




    # Return the total possible points (not including possible points for one dropped homework)
    # Assume all homeworks are worth 10 points and all exams are worth 50 points
    #
    def getAdjustedTotalPossiblePoints(self):
         return 10*(len(self.homeworkScores)-1) + 50*len(self.examScores)

    # Return an integer (not a float!) between 0 and 100 representing the
    # adjusted points scored as a percentage of the adjusted possible total, rounded
    # to the nearest whole number
    # E.g. for total points of 821 out of 1000 possible, the function would return 82
    #
    def getAdjustedTotalPercentage(self):
```

For example, given this test function:

```
def testStudentInfo():
    s = StudentInfo("jim", 123)
    s.addHomeworkScore(5)
    s.addHomeworkScore(6)
    s.addHomeworkScore(8)
    s.addHomeworkScore(2)
    s.addExamScore(45)
    s.addExamScore(35)
    print "Adjusted homework total:", s.getAdjustedHomeworkTotal()
    print "Adjusted homework average:", s.getAdjustedHomeworkAverage()
    print "Exam total:", s.getExamTotal()
    print "Adjusted total points:", s.getAdjustedTotalPoints(), \
        " out of possible ", s.getAdjustedTotalPossiblePoints()
    print "Adjusted total percentage:", s.getAdjustedTotalPercentage()
```

testStudentInfo() produces:

```
>>> testStudentInfo()
Adjusted homework total: 19
Adjusted homework average 6.33333333333
Exam total 80
Adjusted total points: 99  out of possible  130
Adjusted total percentage: 76
```

5. (9 points) Conside the task, like that in Homework 12, of creating a long URL for use with a Google Maps like API. The URL will specify a map center, zoom level, and a number of marker (pin) locations.

   Complete the function generateMapURL(centerLatLon, zoomLevel, tweets) where the inputs are:

   centerLatLon : a 2-element tuple giving the latitude and longitude of the desired center of the map

   zoomLevel : an integer representing the desired map zoom level

   tweets : a list of dictionaries containing Twitter tweet information. Each dictionary might have many keys but you only care about two of them, tweetText and coordinates. For a given tweet dictionary td, the value td['tweetText'] is the text of the tweet and the value of td['coordinates'] is either "NoGeo" or a 2-element tuple, (lat, lon), giving the location of the tweet.

   The output should be a URL of the form:
   `"http://mapapi.cs2110maps.com/map?center=clat,clon&zoom=zoomlevel&markers=lat0,lon0|...|latk,lonk"`
   where clat, clon are the latitude and longitude of the desired map center, zoomlevel is the desired zoomlevel, and lati, loni in the markers list are lats/lons for those tweets that have coordinates associated with them.

   For example, `generateMapURL((43.0, -93.0), 12, [{'tweetText' : "My first tweet", 'coordinates': (43.1, -93.01), 'id' : 23}, { 'coordinates': "NoGeo", 'tweetText': "Wut?"}, {'tweetText': "Done with finals!", 'coordinates': (43.2, -93.0)}])` could return
   `"http://mapapi.cs2110maps.com/map?center=43.0,-93.0&zoom=12&markers=43.1,-93.01|43.2,-93.0"`

   (Note: assume tweet text consists of basic Ascii characters so no special encoding functions are needed.)

```
def generateMapURL(centerLatLon, zoomLevel, tweets):
```

I hope you have a great holiday break!!