Your name: \_\_\_\_\_

## CIS 210 Fall 2013 Final Exam

Write your name at the top of each page before you begin. [5 points]

```
1. [5 points] What does q1() print?

def distinct(txt):
    unique = set()
    count = 0
    for w in txt.split():
        if not (w in unique):
            unique.add(w)
            count += 1
    return count

def q1():
    print( distinct("sauce for the goose sauce for the gander"))
```

2. [5 points] What does q2() print?

def amplify(li, n):
 for i in range(len(li)):
 li[i] = n \* li[i]

def q2():
 x = [3, 4, 5]
 amplify(x,2)
 sum = 0
 for item in x:

sum += item

print(sum)

Your name: \_\_\_\_\_

Page 2 of 6

3. [5 points] What does q3( ) print?

def addlist(a, b):
 for i in range(len(a)):
 if i < len(b):
 a[i] += b[i]

def q3():
 x = [1, 2, 3]
 y = [3, 2, 1]
 z = x
 addlist(x, y)
 sum = 0
 for item in z:
 sum += item
 print(sum)</pre>

Your name:

Page 3 of 6

4. [10 points] For this problem, you may find it useful to draw a simple map or diagram of a swamp. What does q4() print?

```
class Monster:
    def __init__(self, name, x, y, reach):
        self.name = name
        self.xcoord = x
        self.ycoord = y
        self.reach = reach
    def in_reach(self, x, y):
        dx = self.xcoord - x
        if dx < 0:
            dx = 0 - dx
        dy = self.ycoord - y
        if dy < 0:
            dy = 0 - dy
        return self.reach >= dx + dy
class Swamp:
    def __init__(self):
        self.monsters = [ ]
    def add(self,m):
        self.monsters.append(m)
    def safe(self, x, y):
        for m in self.monsters:
            if m.in_reach(x,y):
                return False
        return True
def q4():
    sw = Swamp()
    sw.add( Monster("Grendel", 0,0, 1))
    for x in range(2):
        for y in range(2):
            if sw.safe(x,y):
                print("Safe spot:", x, y)
```

CIS 210 — Fall 2013 Final Exam

class Scores:

5. [10 points] Class Scores is part of a class record for keeping homework scores. Complete the average method.

```
"""Project scores in Snowman Construction 101"""
def __init__(self):
    """New record of scores. None recorded yet"""
    self.project_scores = [ ]
def record(self,score):
    """Record a homework score.
    Args:
        score: integer, the new homework score to record.
    self.project\_scores.append(score)
def average(self):
    """Calculate average (arithmetic mean) of recorded homework scores.
        none (uses previously recorded scores)
    Returns:
        Arithmetic mean of recorded scores, that is,
        sum of scores / number of scores. If no scores
        have been recorded, return 0.
    11 11 11
    # Your code here
```

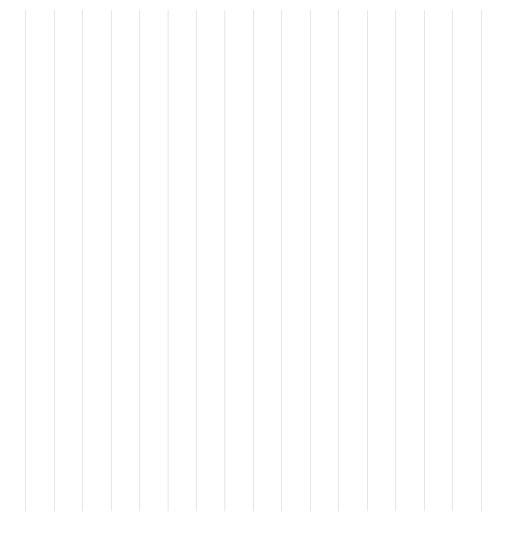
Your name: \_\_\_\_\_

Page 5 of 6

6. [15 points] Finish the function uniform below, consistent with its docstring. You may write additional functions to simplify your code.

```
def uniform(li):
    """Determines whether all the rows of integers in li have the same sum.
    Arguments:
        li: A list of lists of integers
    Returns:
        True if each of the lists in li has the same sum; False otherwise
        Examples:
        uniform( [[50, 50], [100, 0], [25, 25, 25, 25]] ) = True
        uniform( [[ -5, 5, 0 ], [13, -7, -6], [ ]] ) = True
        uniform( [[ 192, 344, 17]] ) = True
        uniform( [] ) = True
        uniform( [[ 7, 3], [5, 2]] ) = False
        uniform( [[17], []] ) = False
        uniform( [[17], []] ) = False
```

### Your code here



CIS 210 — Fall 2013 Final Exam

Your name:

Page 6 of 6

7. [20 points] In the following question, add a number of days to a date. For example, January 15 + 20 days is February 4. For the sake of simplicity we ignore leap year. Finish the function.

```
MONTHS = ["X","Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"]
DAYS_IN_MONTH = [ 0, 31, 28, 31, 30, 31, 30, 31, 30, 31, 30, 31 ]

def days_ahead( start_month, start_day, ahead ):
    """Calculate start date + days ahead (ignoring leap year).
    Args:
        start_month: integer month number, 1..12
        start_day: integer day of month, 1..DAYS_IN_MONTH[start_month]
        ahead: integer, number of days ahead
    Returns: string consisting of 3-letter month name and day, which is start month
        and day + ahead. (We treat February as always having 28 days.)
    Examples:
        days_ahead(1, 31, 0) = "Jan 31"
        days_ahead(1, 15, 20) = "Feb 4"
        days_ahead(11, 20, 90) = "Feb 18"
    """
```

# Your code here

(score)