Test 2: Compsci 06

Owen Astrachan Robert Duvall

November 17/18, 2010

Name: _____

NetID/Login: _____

Honor code acknowledgment (signature) _____

	value	grade
Problem 1	12 pts.	
Problem 2	6 pts.	
Problem 3	12 pts.	
Problem 4	8 pts.	
Problem 5	20 pts.	
Problem 6	10 pts.	
TOTAL:	68 pts.	

This test has 10 pages, be sure your test has them all. Do NOT spend too much time on one question — remember that this class lasts 75 minutes. The last page is blank, if you use it make a note for the problem.

In writing code you do not need to worry about specifying the proper `import statements`. Don't worry about getting function or method names exactly right. Assume that all libraries and packages we've discussed are imported in any code you write.

PROBLEM 1 : (***Exlax (12 points)***)

Part A (4 points)

Complete the list comprehension below to create a list from all the strings in the list `compounds`, a list of strings, that have fewer than six characters.

```
[                  for w in compounds                        ]
```

Part B (4 points)

Write a list comprehension that creates a list of all the multiples of 5 greater than 0 and less than 4096, e.g., [5,10,15,...,4095].

Part C (4 points)

Write a list comprehension that creates a list of all the prime numbers less than 10,000. You can use the function `prime` below that returns True if and only if its parameter `num` is a prime number.

```
def prime(num):
    if num == 2: return True
    if num % 2 == 0: return False
    limit = int(math.sqrt(num)) + 1
    for n in range(3,limit):
        if num % n == 0: return False
    return True
```

PROBLEM 2 : (***What's the Point? (6 points)***)

Write the function `minDistance` that returns the minimal distance between two points in a list of points where points are represented as tuples.

For example the code below will print 4.123 which is the distance between `(-6,6)` and `(-7,2)` which are the two points that are closest.

```
points = [(-7, 2), (-6, 6), (1, 2), (4, 5), (9, -6)]
print minDistance(points)
```

Write your code below, you can call `distance`

```
def distance(a,b):
    """
    calculate and return distance between points represented as tuples
    e.g., between (a_0,a_1) and (b_0,b_1)
    """
    return math.sqrt((a[0]-b[0])**2 + (a[1]-b[1])**2)


def minDistance(points):
    """
    return minimal distance between 2-tuples in points, a list of (x,y) points
    """
```

PROBLEM 3 : (*Comprehensive Care (12 points)*)

In lab this semester we used stock data such as the following. Each element in this list is a three-tuple containing a string representing a date, the closing (float) price of the stock in dollars, and an int that's the number of shares sold. The list below shows three such values for February 1, 2008, January 31, 2008, and Januray 30, 2008.

```
[('2008-02-01', 515.99, 17600500),
 ('2008-01-31', 564.03, 14871300),
 ('2008-01-30', 548.27, 7939600)
]
```

The list comprehension below is used to determine the total number of shares traded for the stocks represented in a list named `stocks`. This expression is shown to remind you how the stock information is accessed. If `stocks` is the list above, this expression returns $17600500 + 14871300 + 7939600 = 40411400$

```
sum([data[2] for data in stocks])
```

Part A (4 points)

Write an expression or code to calculate the total of all dollars spent trading the stocks whose data is stored in `stocks`. For each day, the dollars spent is determined by multiplying the closing price by the number of shares traded. This value should be stored in a variable named `total`. For the example above this total is

$$515.99 \times 17600500 + 564.03 \times 14871300 + 548.27 \times 7939600 = 21822585826.0$$

Part B (4 points)

Write an expression or code to store in `highPrice` the highest closing price for all the data stored in a list `stocks`. You may find it useful to call the function `max` that calculates the maximal value in a list, e.g., `max([1,5,3,2,])` is 5.

Part C (4 points)

Write an expression or code to create a list of the dates (strings) for each day in which the closing price of the stock is more than $500.00. Store this list in a variable `biggies`.

PROBLEM 4 : (*Aged Worms (8 points)*)

One student writes the code below in the Jotto program. This will be compared to another solution.

```
def processCommon(common):
   global possibleWords, lastGuess
   newPoss = []
   for word in possibleWords:
       if commonCount(lastGuess,word) == common:
           newPoss.append(word)
   possibleWords = newPoss
```

Another student sees this code, but decides to write the following to submit.

```
def processCommon(common):
   global possibleWords, lastGuess
   possibleWords = [w for w in possibleWords if commonCount(lastGuess,w) == common]
```

Part A (2 points)

Which version do you prefer? Why?

Part B (6 points)

For extra credit a student keeps a global set of *characters* (one-character strings) that cannot appear in the word the user is thinking of (the secret word). This set is updated after `processCommon` is called and is stored in the global `badLetters`. Write the function `removeBad` that removes every word from `possibleWords` that contains any one (or more) of the letters in `badLetters`. The function should change the value of `possibleWords` by using `badLetters`. For example, if `badLetters` contains a "t" and a "v" then any word in `possibleWords` that has either a 't' or a 'v' (or both) will be removed.

```
def removeBad():
```

PROBLEM 5 : (*A Static List (20 points)*)

Given a list with an odd number of integer elements write the methods below to return the *mean*, *mode*, and *median* values, respectively.

- The *mean* is the average: the sum of the elements in the list divided by the number of elements. It is a float.
- The *median* is the middle element if the elements are sorted, e.g., in a list of 11 elements, it's the 6th element — it's an int since the list has an odd number of int values.
- The *mode* is the value that occurs most often. Assume that the mode is unique, i.e., there's a single element in the list that occurs more often than any other element — it's an int.

For example, for the list `s = [1,5,3,2,5,7,9]` we would have

1. `mean(s)` is 4.571
2. `median(s)` is 5
3. `mode(s)` is 5

Part A (4 points)

```
def mean(nums):
    """
    return average/mean of int values in nums
    """
```

Part B (6 points)

```
def median(nums):
    """
    return median of values in nums which contains
    int values and len(nums) is an odd number
    """
```

Part C (10 points)

```
def mode(nums):
    """
    returns the mode of the values in nums
    which contains int values
    """
```

PROBLEM 6 : (*Eight Days a Week (10 points)*)

You're hired by Apple to manage the purchases of Beatles tracks. A file of purchases is kept with each title stored with the email addresses of the people who have purchased that track. Three tracks are shown below. The title is the first string on each line, separated from email addresses by a comma. Each email address for a given track is separated by a comma as well

```
Drive My Car,ola@duke.edu,pjl@msn.com
Norwegian Wood,pjl@msn.com,jf@foo.edu
Nowhere Man,pkp@nbc.com,pjl@msn.com,joa@gmail.com
```

Write a function `bbFan` that returns the email address of the biggest Beatles fan – the email address of the person who purchased the most tracks. The function is passed the name of the file storing the data.

```
def bbFan(filename):



    file = open(filename)









    file.close()
```

(nothing on this page)