

Test 1 Practice: Compsci 06

Owen Astrachan

February, 2011

Name: _____ (1 point)

NetID/Login: _____

Honor code acknowledgment (signature) _____

	value	grade
Problem 1	20 pts.	
Problem 2	10 pts.	
Problem 3	10 pts.	
Problem 4	20 pts.	
TOTAL:	60 pts.	

This test has 9 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 : (*Won't You Take Me to?*)

Part A

Write the function `totalBill` used to compute the bill presented to customers at a restaurant. If the number of people dining is six or greater an automatic 18% gratuity should be added to the bill, otherwise the bill is unchanged.

For example:

call	return value
<code>totalBill(80.00,5)</code>	80.0
<code>totalBill(80.00,6)</code>	94.40
<code>totalBill(100.0,10)</code>	118.0

Complete the function below.

```
def totalBill(bill, customers):  
    """  
    bill is a float, customers an int, return  
    the total to charge based on these values  
    """
```

Part B

The area of a triangle with sides whose lengths are a, b, c can be computed using *Hero's formula* as

$$\sqrt{s * (s - a) * (s - b) * (s - c)}$$

whwere s is the *semi-perimeter* or $\frac{(a+b+c)}{2}$. Complete function `triangleArea` using the formula to return the area of a triangle whose sides are given by the float parameters a, b, c

```
def triangleArea(a,b,c):  
    """  
    return area of triangle whose sides are floats a,b,c  
    """
```

Part C

Write the function `allDifferent` so that it returns `True` if three int parameters are all different and `False` otherwise. For example:

call	return value
<code>allDifferent(2,3,4)</code>	<code>True</code>
<code>allDifferent(4,3,4)</code>	<code>False</code>
<code>allDifferent(4,4,4)</code>	<code>False</code>
<code>allDifferent(3,2,1)</code>	<code>True</code>

```
def allDifferent(a,b,c):  
    """  
    returns True if a,b,c are all different  
    and returns False otherwise (a,b,c are int values)  
    """
```

Part D

Write the function `lastFirst` described below that returns a String in the format `first last` given a String in the format `last, first` as shown in the examples.

call	return value
<code>lastFirst("Smith, John")</code>	<code>"John Smith"</code>
<code>lastFirst("Van Doren, Mamie")</code>	<code>"Mamie Van Doren"</code>
<code>lastFirst("Begley Jr., Ed")</code>	<code>"Ed Begley Jr."</code>

The String parameter `name` passed to `lastFirst` will always contain a String in the format *"last, first"* where the only comma in `name` comes after the last name, the comma is followed by one space, and then the first name.

```
def lastFirst(name):  
    """  
    returns "first last" given String name in format "last, first"  
    """
```

PROBLEM 2 : (*Absolutely Positively*)

Part A

Complete the function `posCount` so that it returns the number of positive (greater than zero) elements in its list parameter. For example:

call	return value
<code>posCount([-1,0,1,2])</code>	2
<code>posCount([3,4,1,2])</code>	4
<code>posCount([-1,0,-21,0])</code>	0

Complete the function below:

```
def posCount(nums):
```

Part B

Complete the function `odds` that returns a new list consisting of just the odd numbers in its parameter which is a list of numbers. The numbers in the returned list should be in the same order as they are in the parameter `nums`. For example:

call	return
<code>odds([1,3,2,4,6,7])</code>	<code>[1,3,7]</code>
<code>odds([7,5,3,2,9])</code>	<code>[7,5,3,9]</code>
<code>odds([2,4,6])</code>	<code>[]</code>

Complete the function below:

```
def odds(nums):
```

PROBLEM 3 : (*Mean Streets*)

A file contains scores for the students in a class, each student's scores are on one line separated by commas with no spaces. The student name precedes the scores and is separated from the scores by a comma. For example, a data file *scores.csv* might look like this:

```
Alice N Wonderland,10,20,30,40
Jack B. Nimble,50,50,50,0
e.e. cummings,100,90
Sting,90,90,80,80,80
```

Complete the program below that reads the file and prints one line for each line in the data file. Each line printed contains a student's name followed by that student's average. For example, the output for the data file *scores.csv* above is given below.

```
Alice N Wonderland 25.0
Jack B. Nimble 37.5
e.e. cummings 95.0
Sting 84.0
```

Complete the program in the module **Average.py** shown below by filling in missing code in the function **printAverages** so that the module will print the output described above given the data file *names.csv* as shown. The module is given on the next page.

```

def str2ints(strlist):
    """
    strlist is list of strings representing numbers
    return list of ints that correspond to the strings
    """
    return [int(st) for st in strlist]

def average(nums):
    """
    returns average of numbers in nums, a list of integers
    """
    total = sum(nums)
    return total*1.0/len(nums)

def printAverages(filename):
    file = open(filename)
    for line in file:
        data = line.split(",")
        name = data[0]
        nums = data[1:]

        # YOU WRITE CODE HERE

    file.close()

def main():
    printAverages("scores.csv")

if __name__ == "__main__":
    main()

```

PROBLEM 4 : (*Unique New York*)

The Python module/code below shows three functions that are each intended to return `True` if their parameter is a number with no repeated digits and `False` if their parameter is a number with repeated digits. The number 1234 has no repeated digits, the number 1231 has the digit 1 repeated. The output from executing the module is shown after the module.

```
def uniqueOne(n):
    digits = "0123456789"
    string = str(n)
    for d in digits:
        if string.count(d) > 1:
            return False
    return True

def uniqueTwo(n):
    counts = [0]*10
    string = str(n)
    for d in string:
        counts[int(d)] += 1
    return counts.count(1) == len(string)

def uniqueThree(n):
    string = str(n)
    for d in string:
        if string.find(d) != string.rfind(d):
            return False
    return True

funList = [uniqueOne, uniqueTwo, uniqueThree]
nums = [12345, 11245, 10241, 10031, 9876543210]
for fun in funList:
    print "calling ", str(fun), ': '
    for n in nums:
        print n, fun(n)
```

Output is here:

```
calling <function uniqueOne at 0x7f3d2f80c8c0> :
12345 True
11245 False
10241 False
9876 True
calling <function uniqueTwo at 0x7f3d2f80fd70> :
12345 True
11245 False
10241 False
9876 True
calling <function uniqueThree at 0x7f3d2f813b18> :
12345 True
11245 False
10241 False
9876 True
```

Part A

In words (be brief) explain how the output is generated by the lists and loops that appear after the functions `uniqueOne`, `uniqueTwo`, and `uniqueThree`.

Part B

In words (be brief again) explain how each of the functions `uniqueOne`, `uniqueTwo`, and `uniqueThree` works. Don't explain the Python code, explain the general idea behind the logic each of the functions uses to determine if a number has repeated digits.

Part C

Which of the functions will return `True` if the parameter passed to it is a string with no repeated letters and `False` otherwise, e.g., returns `True` for the string `"duplicate"` and `False` for the string `"repeated"` (since the latter contains more than one 'e'). At least one function will work. Justify your answer.