PROBLEM 1 : (*Myrmecophagous?* 24 points)

Write Python statement(s)/expression(s) to solve each of the problems below.

Consider the list `lista` below used to illustrate each problem.

```
lista = ["sloth", "aardvark", "pangolin", "pangolin", \
         "aardvark", "sloth", "sloth", "numbat","anteater"]
```

This list is used to illustrate the problems, but the code you write should work with any values stored in `lista`, don't write code that depends on any particular values stored in the list.

Part A (4 points)

Write Python code that stores in variable `uniq` the number of different values in `lista` — this is five in the example above since the five different strings in `lista` are 'sloth', 'aardvark', 'pangolin', 'numbat', 'anteater'.

Part B (4 points)

Write Python code that stores in variable `smalls` a list of of the strings in `lista` that have fewer than six letters in them. This would be `["sloth", "sloth", "sloth"]` in the example above. The words in `smalls` should be in the same order they appear in `lista`.

Part C (4 points)

Write Python code that stores in variable `most` the number of times the most frequently occurring string in `lista` occurs — this is three in the example above (for `"sloth"`).

Part D (12 points)

Write Python code that stores in variable `ordered` a list of the unique strings in `lista` in order from most frequently occuring to least frequently occuring. Ties should be broken alphabetically, e.g., `"aardvark"` appears before `"pangolin"` in `ordered` below (using `lista` as above) because they both occur twice but `"aardvark"` comes before `"pangolin"` alphabetically. Using `lista` above the values stored in `ordered` are:

```
ordered == ["sloth", "aardvark", "pangolin", "anteater", "numbat"]
```

because the number of occurrences of each of these is 3, 2, 2, 1, and 1, respectively. Note that `"anteater"` is alphabetically before `"numbat"` and both occur one time. (You'll earn more than half-credit if strings are ordered correctly by number of occurrences, but you don't break ties alphabetically.)

Part D (8 points)

Program

The program below generates the output on the right when run, showing that each simulated dice roll (a,b) occurs roughly the same number of times. However, the number of times each *sum* is rolled is different since there is only one way to roll a two: (1,1), but six ways to roll a seven: (1,6), (2,5), (3,4), (4,3), (5,2), (6,1). Write the function `get_totals` that returns a dictionary in which the key is a number from 2-12, inclusive, representing the sum of rolling two simulated dice; the corresponding value is the number of times the total occurs. The parameter to `get_totals` is the dictionary returned by `track_rolls`.

```
import random

def get_roll():
    return (random.randint(1,6), random.randint(1,6))

def track_rolls(repeats):
    d = {}
    for x in range(0,repeats):
        roll = get_roll()
        if roll not in d:
            d[roll] = 0
        d[roll] += 1

    for key in sorted(d.keys()):
        print key,d[key]

    return d

def main():
    d = track_rolls(10000)

if __name__ == "__main__":
    main()
```

For example, adding the line `p = get_totals(d)` in the function `main` and printing the contents of `p` should result in the output below given a dictionary storing information as shown on the right (the output won't necessarily be sorted by sum):

```
2 263
3 575
4 875
5 1121
6 1386
7 1627
8 1299
9 1152
10 888
11 531
12 283
```

(write code on next page)

Output from Running Program

```
(1, 1) 263
(1, 2) 283
(1, 3) 289
(1, 4) 261
(1, 5) 293
(1, 6) 270
(2, 1) 292
(2, 2) 297
(2, 3) 269
(2, 4) 297
(2, 5) 270
(2, 6) 254
(3, 1) 289
(3, 2) 284
(3, 3) 272
(3, 4) 245
(3, 5) 242
(3, 6) 295
(4, 1) 307
(4, 2) 246
(4, 3) 262
(4, 4) 273
(4, 5) 308
(4, 6) 302
(5, 1) 278
(5, 2) 301
(5, 3) 261
(5, 4) 254
(5, 5) 285
(5, 6) 278
(6, 1) 279
(6, 2) 269
(6, 3) 295
(6, 4) 301
(6, 5) 253
(6, 6) 283
```

```
def get_totals(rolld):
    """
    rolld is a dictionary in which (a,b) tuples are
    the keys, the corresponding value is the number of times
    (a,b) was rolled in a dice simulation. Return dictionary
    in which keys are unique values of a+b for (a,b) in
    rolld and value is number of times sum a+b occurs for
    each key
    """
```

PROBLEM 3 : (*Follow the Money (18 points)*)

A list of political contributions in 2010 is stored in a Python list of tuples named `contribs` where each tuple stores four values: a string, the politician's name; a two-letter string, a US State abbreviation; an integer, the total of all donations to the politician; and a single letter 'R', 'D', or 'I' for Republican, Democrat, or Independent, respectively.

For example, the second line below shows that *Barbara Boxer* from CA (California) received $20,314,189 in donations and she is a Democrat.

```
contribs = [
   ("Jeff Greene","FL",23807119,"D")
   ("Barbara Boxer","CA",20314189,"D")
   ("Charles E Schumer","NY",17302006,"D")
   ("Harry Reid","NV",17213358,"D")
   ("Kirsten Elizabeth Gillibrand","NY",12900217,"D")
   ("Joseph A Sestak Jr","PA",11842844,"D")
   ("Linda Mcmahon","CT",46682270,"R")
   ("Sharron E Angle","NV",21470516,"R")
   ("John S Mccain","AZ",20077490,"R")
   ("Marco Rubio","FL",18251722,"R")
   ("Carly Fiorina","CA",17935605,"R")
   ("Scott P Brown","MA",17527893,"R")
]
```

As an example, to find the total of all contributions to all politicians we could use this Python expression:

```
total = sum([c[2] for c in contribs])
```

Part A (4 points)

Write a Python expression or code to store in variable `rtotal` the total of all donations for all politicians who are Republicans.

Part B (4 points)

Write a Python expression or code that creates a list of strings: the names of all politicians who have received more than $15 million in donations, store this list in variable `heavies`.

Part C (10 points)

For this part of the problem new data is provided in addition to the list `contribs` in the previous parts.

Each of the politician's for whom data is found in the list `contribs` is the key in a dictionary `donors` whose value is a list of tuples, the tuples giving the contributions for each donor to that politician. For example, for Senator Rob Portman of Ohio part of his dictionary entry is shown below:

```
"Rob Portman" : [("L. Abbott", "4/22/10", 1900), ("V. Alpaugh", "12/29/09", 1000), ...
                 ("C. Klein", "11/23/10", 500)]
```

The tuples in the list of donors have three values: the name of the donor, the date of the donation, and the amount of money donated. In the data shown above, C. Klein gave $500.00 (to Rob Portman) on November 23, 2010 (`"11/23/10"`).

Write the function `small_donors` that returns a list of two-tuples. The first element of each two-tuple is the name of a politician that is in the list `contribs` (the first element of the tuples in list `contribs`, see previous page) the second element in the two-tuple is an integer, the number of *small* donors, those who gave less than $1000 to the candidate. For example, the list returned might look like this depending on the data passed to `small_donors`:

```
[("Rob Portman", 52), ("Carly Fiorina", 107), ... ("Barbara Boxer", 972)]
```

The tuples in the list can be in any order. Every politician in `contribs` is a key in `donors`.

```
def small_donors(contribs, donors):
    """
    contribs is a list described earlier containing 4-tuples,
    in which first element is politician's name

    donors is a dictionary: key is politician's name and value
    is list of 3-tuples as above (name,date,money)

    return list of 2-tuples (politician's name, #small donors)
    """
```

PROBLEM 4 : (*Top Songs (10 points)*)

Rolling Stone magazine published a list of the top 500 songs of all time in 2004 and updated the list in 2010. A file stores the song, the artist, and the year the song was released as shown below.

```
Like a Rolling Stone:Bob Dylan:1965
(I Can't Get No) Satisfaction:The Rolling Stones:1965
Imagine:John Lennon:1971
What's Going On:Marvin Gaye:1971
...
Born to Run:Bruce Springsteen:1975
Help!:The Beatles:1965
```

Write the function `artists` that returns a dictionary in which the key is an artist (group, singer) and the corresponding value is a list of the song titles from that artist. For example, both of these entries would appear in the dictionary returned:

```
"The Beatles" : ["Hey Jude", "Yesterday", "I Want to Hold Your Hand", "Help!", ...]
"The Rolling Stones" : ["(I Can't Get No) Satisfaction", "Sympathy for the Devil", ...]
```

The parameter `filename` is the name of a file as shown above. Return the dictionary described.

```
def artists(filename):
    """
    return dictionary in proper format given parameter
    filename which has song information in proper format
    """


    f = open(filename)









    f.close()
```

PROBLEM 1 : (*IMDB*)

The Internet Movie Data Base (IMDB) company has asked you to help them determine the actor who has appeared in the most movies. IMDB data is stored by movie in a CSV format with the movie title first and actors following the title:

```
Taxi Driver,Robert De Niro, Jodie Foster, Cybill Shepherd, Harvey Keitel
Contact,Jodie Foster, Tom Skerritt, Matthew McConaughey,Max Martini
Sommersby,Richard Gere,Jodie Foster,Bill Pullman,James Earl Jones
```

Write a python function named `topActors` that takes the name of the data file as a parameter and prints the top five actors in terms of movie-appearances and their movies in the format shown below. The order of the movies printed does not matter. Assume there will be data for more than five different actors in the file and do not worry about breaking ties in terms of number-of-movies for an actor.

```
Jodie Foster: 10 movies
   Taxi Driver
   Panic Room
   Little Man Tate
   Anna and the King
   Contact
   Sommersby
   Silence of the Lambs
   Inside Man
   The Accused
   The Hotel New Hampshire
Richard Gere: 5 movies
   Pretty Woman
   Primal Fear
   Sommersby
   Nights in Rodanthe
   Brooklyn's Finest
James Earl Jones: 3 movies
   Sommersby
   Star Wars: A New Hope
   Dr. Strangelove
```

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 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()
```