PROBLEM 1 : (***What is the output? (18 points)***)

Part A. What is the output of the following code segments?
Write the output to the right. Note that there is only output for the print statements.

```
lista = ['S']
lista.append('A')
print lista
lista.append( ['T','R'] )
print lista
lista = ['G', 'B', 'W']
lista.insert(1,'U')
lista.insert(2,'C')
print lista
lista = ['C', 'S', 'U', 'B']
lista.pop()
print lista
lista.remove('S')
print lista
# -------------------------
seta = set([7,7,4,8,7,4,8,4])
seta.add(4)
print sorted(list(seta))
seta.add(3)
print sorted(list(seta))
seta = set([7,7,4,8,7,4,8,4])
seta.remove(4)
print sorted(list(seta))
# -------------------------
seta = set([9, 2, 1])
setb = set([8, 2, 9])
print seta^setb
print setb.union(seta)
print seta&setb
# -------------------------
d = {'F':4, 'H':3, 'K':4, 'A':2}
print sorted(d.keys())
d['H'] = 5
d['P'] = 7
print sorted(d.keys())
print sorted(d.values())
print sorted(d.items())
```

```
OUTPUT
------
```

Part B. What is the output of the following code segment? **Write the output after the code segment**. Note that there is only output for the print statements.

```
nums = [10, 70, 10, 50, 70, 70, 10, 50, 70, 40]
dictnums = {}
for n in nums:
    if n not in dictnums:
        dictnums[n] = 0
    dictnums[n] += 1
print dictnums.keys()
print dictnums.values()
print max(dictnums.values())
```

What is the output?

PROBLEM 2 : (*Short Code/Answer (16 points)*)

For parts that involve writing code, your code should also work if the given list was modified with different values.

A. (3 pts) Consider the following code.

What is the value of `result` after this code executes?

```
words = ['car', 'airplane', 'train', 'scooter', 'bike', 'skates']
result = [w for w in words if w.count('a') > 0]
```

B. (3 pts) Consider the following code.

What is the value of `result` after this code executes?

```
phrase = "compsci econ english math french spanish"
result = [w[0] + w[-1]  for w in phrase.split() if len(w) > 6]
```

C. (4 pts) **Write one line of code that includes a list comprehension** to assign to the variable `result` a list for the following problem. Given a phrase of words, create a list that contains a new word for all the words whose length is evenly divisible by 3. Each new word is the first three letters twice of the original word. The words in phrase are all lowercase letters and separated by a blank. The resulting list should have the new words in the same order as the original list.

```
phrase = 'squirrel fox deer armadillo raccoon turtle cat'
```

For example, if `phrase` is the string above, then after executing the list comprehension, then `result` would be the list `['foxfox', 'armarm', 'turtur', 'catcat']`

Write the list comprehension below.

```
result =
```

D. (6 pts) Write a function named `howMany` that has one parameter `fruitlist` that is a list of strings. Each string represents a person and the type of fruit they are bringing to an event. The format of each string is a name (one word), a colon and a fruit (one word). This function returns a list of ordered tuples. For each unique name a tuple is created. The first part of the tuple is the name and the second part is the number of different types of fruit the person is bringing. Each fruit appears only once in the list. The list of tuples returned are **sorted** by names.

For example, suppose `fruitlist` has the following value:

```
fruitlist = ['Smith:apples', 'Hopper:blueberries', 'Smith:oranges',
     'Hopper:grapes', 'Grishom:plums', 'Krishna:pineapples', 'Hopper:figs']
```

Then the call `howMany(fruitlist)` would return the list

`[('Grishom', 1), ('Hopper', 3), ('Krishna', 1), ('Smith', 2)]`

```
def howMany(fruitlist):
```

PROBLEM 3 : (*Getting that well-paid job (14 points)*)

Consider the following data file of information on students getting job offers. Each line in the file represents a job offer from a company to a student. That offer would include salary and possibly a bonus and moving budget. Here is the format of a line. The first item is the name of the company (one or more words), followed by colon, followed by student's id number, followed by colon, followed by the salary. There could be one or two additional numbers for a bonus and moving budget. If so, then there is a colon and a number for the bonus. If there is a moving budget, then the bonus is followed by a colon and a number for the moving budget.

An example of the data file is shown below. For example, in the first line, the company is Google, the student id is 21023, the salary is 65000, there is bonus of 5000 and there is a moving budget of 3000. In the second line, the company is Two Sigma, the student id is 26853, and there is a salary of 70000. There is no bonus or moving budget. In the third line the company is Paypal, the student id is 21023, the salary is 60000 and there is a bonus of 10000. This is a second offer for the student with id 21023.

```
Google:21023:65000:5000:3000
Two Sigma:26853:70000
Paypal:21023:60000:10000
Capital One:26489:55000:5000:4000
Google:28311:62000:5000:2000
Microsoft:26853:75000:3000
Bloomberg:21023:68000:8000
LinkedIn:26489:62000
State Farm:21023:68000
Capital One:28311:70000:0:2000
Google:26489:62000:8000:4000
Capital One:24712:56000:4000
```

A. (6 pts) Write the function `setup` that has one parameter `filename` which represents the name of the file. This function returns a list of lists of three to five items, where each list represents one offer to a student. Each list will have the company name, the student id and the salary. It may also have a bonus or moving budget. The third through fifth items should be integers.

For example, the line `data = setup("offerData.txt")` where "offerData.txt" is the file above would result in data having the value on the next page.

```
data = [ ['Google', '21023', 65000, 5000, 3000],
  ['Two Sigma', '26853', 70000],
  ['Paypal', '21023', 60000, 10000],
  ['Capital One', '26489', 55000, 5000, 4000],
  ['Google', '28311', 62000, 5000, 2000],
  ['Microsoft', '26853', 75000, 3000],
  ['Bloomberg', '21023', 68000, 8000],
  ['LinkedIn', '26489', 62000],
  ['State Farm', '21023', 68000],
  ['Capital One', '28311', 70000, 0, 2000],
  ['Google', '26489', 62000, 8000, 4000],
  ['Capital One', '24712', 56000, 4000] ]
```

Complete the function setup below.

```
def setup(filename):
    f = open(filename)
```

B. (8 pts) Write the function `offers` that has two parameters, `data`, and `student`, where `data` is the list of lists in the format resulting from Part A, and `student` is the student id for a student.

This function computes a list of tuples of the offers this student has received, where the first item in the tuple is the company giving the offer, and the second item is an integer which is

the sum of the salary, bonus (if any) and moving budget (if any). The tuples are ordered by increasing order of total compensation.

For example, assume *data* is the lists of lists of items on the previous page. The result of calling `offers(data, "21023")` is the four offers the student with id "21023" received: `[('State Farm', 68000), ('Paypal', 70000), ('Google', 73000), ('Bloomberg', 76000)]`. The result of calling `offers(data, "26489")` is `[('LinkedIn', 62000), ('Capital One', 64000), ('Google', 74000)]`. In both note the tuples are ordered by increasing compensation.

```
def offers(data, student):
```

PROBLEM 4 : (*Club Membership (44 points)*)

Suppose you have data about students and the clubs they participate in, stored in a list of lists, where each list represents a student and one club they are in. In particular, each list has **five strings**: the name of the club (one or more words), the last name of the student (one word), the student's graduation year, the number of club meetings they have attended for this club, and a string from ['1', '2', '3', '4', '5'] representing the level of leadership in the club (with '5' the highest).

For example, suppose `data` is the list below. The first list in `data` represents the student 'allen' who is in the 'chess beginner' club. This student will graduate in '2018', has attended 10 meetings for this club, and has a low level of leadership, '2'. The second list represents the student 'smith' who is in the 'photography' club. This student will graduate in '2020', has attended 10 meetings for this club, and is in a high level of leadership for the club, a '5'. Note a student can be in more than one club and a club can have more than one student. Student names are unique in that no name has different graduation years. For example, there is only one 'smith' and they graduate in '2020'.

```
data = [ ['chess beginner', 'allen', '2018', '10', '2'],
['photography', 'smith', '2020', '10', '5'],
['chess advanced', 'smith', '2020', '7', '1'],
['photography', 'allen', '2018', '5', '1'],
['juggling', 'swift', '2020', '12', '3'],
['soccer advanced', 'zhou', '2021', '24', '4'],
['chess advanced', 'zhou', '2021', '8', '3'],
['soccer beginner', 'martin', '2018', '18', '5'],
['photography', 'alford', '2019', '3', '1'],
['chess beginner', 'taylor', '2018', '12', '1'],
['soccer beginner', 'smith', '2020', '16', '2'],
['juggling', 'park', '2019', '14', '3'],
['soccer beginner', 'alford', '2019', '18', '1'],
['juggling', 'smith', '2020', '15', '1'],
['photography', 'huang', '2021', '5', '1'],
['juggling', 'taylor', '2018', '14', '5'],
['chess beginner', 'wang', '2020', '12', '4'],
['chess beginner', 'bazil', '2019', '12', '5'],
['juggling', 'alford', '2019', '10', '2'] ]
```

In writing any of these functions, **you may call any other function you wrote for this problem.** Assume that function is correct, regardless of what you wrote.

A. (7 pts) Write the function named `clubs` which has one parameter, `data`, that is a nonempty list of lists of five strings in the format mentioned earlier. This function returns a list of the **unique, sorted** names of the clubs in `data`.

For example, if `data` is the example list of lists mentioned earlier then the

call `clubs(data)` would return the list `['chess advanced', 'chess beginner', 'juggling', 'photography', 'soccer advanced', 'soccer beginner']`

```
def clubs(data):
```

B. (7 pts) Write the function `studentsClubs` which has two parameters, `data`, that is a nonempty list of lists of five strings in the format mentioned earlier, and an integer `num` representing a leadership level with value from 1 to 5. This function returns a **sorted unique** list of students that have a leadership level of num or higher in at least one club.

For example, if `data` is the list of lists mentioned earlier, then `studentsClubs(data, 4)` would return the list `['bazil', 'martin', 'smith', 'taylor', 'wang', 'zhou']` All of these students have leadership 4 or higher in some club. Note the resulting list of students is unique and in sorted order.

```
def studentsClubs(data, num):
```

C. (7 pts) Write the function `clubsCommon` which has two parameters, `data`, that is a nonempty list of lists of five strings in the format mentioned earlier, and a list named `clist` that is a list of clubs at another university. This function returns a **sorted unique** list of those clubs that are in common between `clist` and in data.

For example, if `data` is the list of lists mentioned earlier and if `clist` is the list of clubs `['hacking', 'photography', 'outdoors', 'juggling']` then the call `clubsCommon(data, clist)` would return the list `['juggling', 'photography']` .

```
def clubsCommon(data, clist):
```

D. (7 pts) Write the function `dictNamesToMeetings` which has one parameter, `data`, that is a nonempty list of lists of five strings in the format mentioned earlier. This function returns a dictionary where each student is mapped to a list of tuples, the first item in the tuple is the name of a club the person is in and the second item in the tuple is an integer representing the number of meetings this person attended for this club.

For example, if data is the list of lists mentioned earlier, then the call `dictNamesToMeetings(data)` would return a dictionary with several entries. One entry would have key `'taylor'` with value `[('chess beginner', 12), ('juggling', 14)]`.

```
def dictNamesToMeetings(data):
```

E. (8 pts) Write the function `nameMostMeetings` which has one parameter, `data`, that is a nonempty list of lists of five strings in the format mentioned earlier. This function calculates the person with the largest total meeting attendance, and returns a tuple of two items, which is the name of the person and the list of clubs they are a member of, in sorted order. Assume there is no tie.

For example, if data is the list of lists mentioned earlier, then the call `nameMostMeetings(data)` would return the tuple `('smith', ['chess advanced', 'juggling', 'photography', 'soccer beginner'])` , since the total number of meetings for smith is the largest such number (7+16+15+10 = 48).

```
def nameMostMeetings(data):
```

F. (8 pts) Write the function `studentMost` which has one parameter, `data`, that is a nonempty list of lists of five strings in the format mentioned earlier. This function returns the name of the student that is in the most clubs. If there is a tie, then return any such student that ties. **You must build a dictionary as part of solving this problem.**

For example, if `data` is the list of lists mentioned earlier, then the call `studentMost(data)` would return the student `'smith'`, who is in four clubs, the most number of any of the other people.

```
def studentMost(data):
```