PROBLEM 1 : (***What is the output? (16 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.

OUTPUT

```
lista = ['A', 'R']
listb = ['S', 'Y']
lista.extend(listb)
print lista
lista = ['T', 'A', 'Q', 'A']
lista.insert(2,'H')
print lista
lista.pop()
print lista
lista.remove('A')
print lista
# -------------------------
seta = set([3,1,3,2,4,2,1,3])
seta.add(4)
seta.add(6)
print sorted(list(seta))
seta = set([3,1,3,2,4,2,1,3])
seta.remove(3)
print sorted(list(seta))
# -------------------------
seta = set([9, 3, 4, 7])
setb = set([4, 1, 7])
print setb.symmetric_difference(seta)
print setb.union(seta)
print seta&setb
# -------------------------
lista = ['N', 'L', 'H', 'J']
d = {'N':2, 'A':7, 'L':5, 'D':2}
print sorted(d.keys())
print [d[i] for i in lista if i in d]
d['X'] = 5
d['D'] = 6
print sorted(d.keys())
print sorted(d.values())
```

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.

```
colors = ['orange', 'blue', 'rose', 'plum', 'purple', 'black', 'pink']
dict = {}
for item in colors:
    if len(item) not in dict:
        dict[len(item)] = 1
    else:
        dict[len(item)] += 1
print dict.keys()
print dict.values()
print max(dict.values())
```

What is the output?

PROBLEM 2 : (*Short Code/Answer (17 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 temp after this line executes?

```
values = [9, 13, 17, 22, 6, 12]
temp = [n for n in values if n%3>0]
```

B. (3 pts) Consider the following code.

What is the value of temp after this line executes?

```
words = ['yellow', 'purple', 'green', 'blue']
temp = sorted([words[i]*i for i in range(len(words)) if i>1])
```

C. (5 pts) **Write one line of code that includes a list comprehension** to assign to the variable `temp` a list of the strings from the list `words` in which each word from the original list that is of length four or more has the first and last letter swapped. The resulting list should have the modified words in the same order as the original list.

```
words = ['yellow', 'purple', 'green', 'blue', 'red', 'pink']
```

For example, if `words` was the list above, then after executing the list comprehension, then `temp` would be the list `['welloy', 'eurplp', 'nreeg', 'elub', 'kinp']`

Write the list comprehension below.

```
temp =
```

D. (6 pts) Write a function named `namestuff` that has one parameter `names` that is a list of strings representing names. Each name is one or more words separated by a blank. This function returns a list of tuples. For each name that has three or more words, a tuple is created. The first part of the tuple is a string of the first and last words of the name separated by a blank, and the second part of the tuple is a string of the words between the first and last word in the name, with no blanks between them.

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

```
names = ['long john silver', 'mary sue ellen jo cole', 'prince', 'bruce wayne']
```

Then the call `namestuff(names)` would return the list

`[('long silver', 'john'), ('mary cole', 'sueellenjo')]`

Note that only those names with three or more words result in a tuple in the new list, and the tuples appear in the same order the names were in the original list.

```
def namestuff (names):
```

PROBLEM 3 : (*Wins and Losses (15 points)*)

Consider the following data file of information on club basketball teams. Each line in the file represents two teams playing each other and their scores. The format of each line in the file is team1, followed by a hyphen, followed by the number of points team1 made, followed by a colon, followed by team2, followed by a hyphen, and followed by the number of points team2 made. The first team on each line is the home team, where the game was played.

An example of the data file is shown below. For example, in the first line, duke was the home team and duke played against unc, with duke scoring 78 points and unc scoring 76 points, so duke won the game.

```
duke-78:unc-76
unc-87:virginia tech-80
wake forest-73:duke-92
miami-82:unc-79
wake forest-67:miami-77
ncsu-68:unc-70
unc-80:gatech-65
ncsu-77:virginia tech-73
virginia tech-83:wake forest-79
gatech-75:ncsu-81
gatech-81:wake forest-70
duke-76:ncsu-74
virginia tech-75:miami-74
```

A. (7 pts) Write the function `processinfo` that has one parameter `filename` which represents the name of the file. This function returns a list of lists of items in which each inner list has four items and represents one line from the file. The first item is a string of team1's name, the second item is the integer number of points team1 scored, the third item is a string of team2's name, and the fourth item is the integer number of points team2 scored.

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

```
data = [ ['duke', 78, 'unc', 76],
['unc', 87, 'virginia tech', 80],
['wake forest', 73, 'duke', 92],
['miami', 82, 'unc', 79],
['wake forest', 67, 'miami', 77],
['ncsu', 68, 'unc', 70],
['unc', 80, 'gatech', 65],
['ncsu', 77, 'virginia tech', 73],
['virginia tech', 83, 'wake forest', 79],
['gatech', 75, 'ncsu', 81],
['gatech', 81, 'wake forest', 70],
['duke', 76, 'ncsu', 74],
['virginia tech', 75, 'miami', 74] ]
```

Complete the function processinfo below.

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

B. (8 pts) Write the function `schoolsScore` that has three parameters, `data`, `team` and `num`, where `data` is the list of lists in the format from Part A, *team* is a string and *num* is an integer.

This function returns a list of tuples, where each tuple is information about a game that `team` played in in which the winning team had at least num points. Each tuple has the name of the other team, followed by the number of points the winning team had.

For example, assume *data* is the lists of lists of four items on the previous page. The examples below show the result of calling `schoolsScore` with this filename, a team name and an integer. For example, duke played in only one game where the winning team won by 80 or more points, against wake forest. For wake forest, none of their games had a winning team score 95 or more points. For unc, they played in four games in which the winning team scored 75 or more points. They played duke in which the winner had 78 points, they played virginia tech in which the winner had 87 points, they played miami in which the winner had 82 points, and they played gatech in which the winner had 80 points.

call	returns
schoolsScore(data, "duke", 80)	[('wake forest', 92)]
schoolsScore(data, "wake forest", 95)	[]
schoolsScore(data, "unc", 75)	[('duke',78),('gatech',80),('miami',82),('virginia tech',87)]

```
def schoolsScore(data, team, num):
```

PROBLEM 4 : (*Movie Stars (44 points)*)

Suppose you have data about movies and actors in the format of a list of lists, where each list represents the actor in one movie. In particular, each list has **five strings**: the name of a movie, the name of an actor in that movie, the year the movie was released, the total time of the movie in minutes and the total time in minutes this actor appeared in this movie.

For example, suppose `datalist` is the list below. The first list in `datalist` represents the movie "Saving Mr. Banks". The second item is the actor "Tom Hanks", the third item is the year "2016" the movie was released, the fourth item is the total time in minutes for the movie, "125", and the fifth item is the total amount of time Tom Hanks was actually in the movie, "65" minutes.

```
datalist = [
['Saving Mr. Banks', 'Tom Hanks', '2016', '125', '65'],
['Saving Mr. Banks', 'Emma Thompson', '2016', '125', '84'],
['Enough Said', 'James Gandolfini', '2013', '93', '52'],
['Captain Phillips', 'Catherine Keener', '2013', '134', '22'],
['The Da Vinci Code', 'Tom Hanks', '2006', '149', '85'],
['Saving Mr. Banks', 'Colin Farrell', '2016', '125', '25'],
['Forrest Gump', 'Sally Field', '1994', '142', '56'],
['Mrs. Doubtfire', 'Robin Williams', '1993', '125', '94'],
['Captain Phillips', 'Tom Hanks', '2013', '134', '110'],
['Enough Said', 'Catherine Keener', '2013', '93', '21'],
['The Da Vinci Code', 'Ian McKellen', '2006', '149', '60'],
['Hello, My Name is Doris', 'Sally Field', '2015', '95', '84'],
['Alone in Berlin', 'Emma Thompson', '2016', '103', '70'],
['Forrest Gump', 'Tom Hanks', '1994', '142', '110'],
['Mrs. Doubtfire', 'Sally Field', '1993', '125', '45'] ]
```

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 `movies` 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** names of movies from data, in any order.

For example, if `datalist` is the example list of lists mentioned earlier then the call `movies(datalist)` would return the list `['Mrs. Doubtfire', 'The Da Vinci Code', 'Captain Phillips', 'Enough Said', 'Alone in Berlin', 'Saving Mr. Banks', 'Hello, My Name is Doris', 'Forrest Gump']`

```
def movies(data):
```

B. (7 pts) Write the function `actorsTimeIn` which has two parameters, `data`, that is a nonempty list of lists of five strings in the format mentioned earlier, and an integer `time`. This function returns a **sorted unique** list of the actors that appear for `time` minutes or more in some movie.

For example, if `datalist` is the list of lists mentioned earlier, then `actorsTimeIn(datalist, 80)` would return the list `['Emma Thompson', 'Robin Williams', 'Sally Field', 'Tom Hanks']`, the actors that are in some movie for 80 minutes or more. Note the actors are in sorted string order.

```
def actorsTimeIn(data,time):
```

C. (7 pts) Write the function `moviesNotIn` which has two parameters, `data`, that is a nonempty list of lists of five strings in the format mentioned earlier, and a list named `favorite` that is a list of favorite movies. This function returns a list of movies from `favorite` that are not any of the movies in data.

For example, if `datalist` is the list of lists mentioned earlier and if `favorite` is the list of movies `['Hello, My Name is Doris', 'Enough Said', 'Star Trek Beyond']`, then the call `moviesNotIn(data, favorite)` would return the list `['Star Trek Beyond']`.

```
def moviesNotIn(data,favorite):
```

D. (8 pts) Write the function `popularActors` which has one parameter, `data`, that is a nonempty list of lists of five strings in the format mentioned earlier. This function computes the list of the actors that appear in the most movies from datalist. **You must build a dictionary as part of solving this problem.**

For example, if `datalist` is the list of lists mentioned earlier, then the call `popularActors (datalist)` would return the list `['Tom Hanks']` , who is in four movies in datalist, the most number of any of the actors. If there was a tie, then you would return a list of all the actors that tied, in any order.

```
def popularActors(data):
```

E. (7 pts) Write the function `dictMovieToActors` 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 of movies, each movie is mapped to a list of tuples, where each tuple is an actor in that movie, and the total number of minutes the actor was in that movie.

For example, if datalist is the list of lists mentioned earlier, then the call `dictMovieToActors(datalist)` would return a dictionary with several entries. One entry would have key 'Mrs. Doubtfire' with value [('Robin Williams', '94'), ('Sally Field', '45')] as those are the actors in that movie from datalist.

```
def dictMovieToActors(data):
```

F. (8 pts) Write the function `movieMostActorTime` 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 movie that has the most overall total time of actors in that movie. Assume there is no tie.

For example, if datalist is the list of lists mentioned earlier, then `movieMostActorTime(datalist)` would return `'Saving Mr. Banks'`, as that movie has actors with times in that movie of 65 + 84 + 25 for a total of 174, more total actor minutes than any other movie.

```
def movieMostActorTime(data):
```