(Note 001 meets in White Lecture Hall, 002 meets in LSRC B101)

PROBLEM 1 : (*What is the output? (20 points)*)

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
                                                                    ------
sounds = ['do', 're', 'me', 'do', 're', \
           'me', 'fa', 'so', 'so', 'so']
aset = set(sounds)
aset.add('fa')
alist = list(aset)
alist.sort()
print alist

sounds = ['la', 'me', 'la', 'la']
sounds.pop()
print sounds
sounds.append('do')
print sounds

lista = [13, 2, 6, 8, 2]
seta = set(lista)
listb = [2, 4, 6, 8, 10]
setb = set(listb)
print seta.intersection(setb)
print seta.union(setb)
print seta.difference(setb)

dicta = {'UNC':'B', 'Duke':'B', 'ECU':'G', 'WFU':'Y'}
print dicta['Duke']
dicta['ECU'] = 'P'
print dicta.keys()
items = set(dicta.values())
print items

dicta = {'UNC':'B', 'ECU':'G', 'Duke':'B', 'WFU':'Y'}
dictb = {}
for (key,value) in dicta.items():
    dictb[value] = key
keylist = dictb.keys()
keylist.sort()
print keylist
```

PROBLEM 2 : (*List Comprehensions, eh? (11 points)*)

A. (5 pts) Consider the following code.

```
nums = [5, 3, 8, 2, 4]
somelista = [x*3 for x in nums if x % 2 == 0]
print somelista

somelistb = [x for x in nums if x > sum(nums)/len(nums)]
print somelistb

value = min(somelistb)
print value
```

What is the resulting output of this code?

Consider the list `words` for the next two problems.

words = ['diver', 'astronaut', 'geologist', 'clown', 'lifeguard', 'nurse', 'Beth', 'Clarisse', 'Adam']

When asked to write a list comprehension, you can receive **partial credit** for correct code that is not a list comprehension.

B. (3 pts) Write a list comprehension to assign to the variable `startsWithA` a list of strings from `words` that start with the letter a (either upper or lower case), appearing in the same order they appear in the list.

For example, using the list `words` from above, then after executing the list comprehension, then startsWithA would be the list ['astronaut', 'Adam']

Write the list comprehension below.

startsWithA =

C. (3 pts) Write a list comprehension to assign to the variable `firstThree` a list of strings from `words` that are the first three letters of the word if the original word is of length greater than five. These strings should appear in the same order their original words appear in the list.

For example, using the list `words` from above, then after executing the list comprehension, then firstThree would be the list ['ast', 'geo', 'lif', 'Cla']

Write the list comprehension below.

firstThree =

PROBLEM 3 : (*Food a plenty (15 points)*)

A. (7 pts) Consider the following data file in which each line in the file has the name of a food item, followed by a colon, followed by the price of the item, followed by a colon, followed by the name of the person who purchased the food item.

An example of the data file might be:

```
french fries:2.00:John James Smith
chicken salad:12.50:Surly Bo Williams
hamburger:6.50:John James Smith
chicken sandwich:5.25:Lisa Glover
coke:2.50:Surly Bo Williams
chocolate milkshake:3.00:Lisa Glover
diet coke:2.50:John James Smith
```

Write the function `fileToList` that has one parameter `fileMeals` which represents a file that is already open and ready for reading. This function returns a list of lists of strings in which each inner list is three strings. The first string is the food item, the second string is the price of the item, and the third string is the name of the person who purchased the food item.

For example, `fileToList(restaurant)` where restaurant is the file above that is open and ready to read would return the list (not all parts shown):

```
[['french fries', '2.00', 'John James Smith'],
  ['chicken salad', '12.50', 'Surly Bo Williams']
  ... rest not shown ..
]
```

Complete the function fileToList below.

```
def fileToList(fileMeals):
# Assume file is open for reading. Read in data from file
# and return in this format: list of lists,
# each list has three strings representing food item, price and person
```

B. (8 pts) Write the function `foodEaten` that has two parameters, `data`, where data is a list of lists of strings in the format from part A, and `name` is the **last name (one word)** of a person. Note the full name of the person is in the data file, and the last name is the last word in the full name. This function returns a list of the food items this person bought.

For example, assume *fooditems* is the resulting lists of lists from part A based on the file from part A. In the first example, 'John James Smith' is the only person with the last name 'Smith' and he bought the three items 'french fries', 'hamburger' and 'diet coke'. In the second example, 'Lisa Glover' bought two items 'chicken sandwich' and 'chocolate milkshake'.

```
def foodEaten(data, name):
```

call	returns
foodEaten(fooditems, 'Smith')	['french fries', 'hamburger', 'diet coke']
foodEaten(fooditems, 'Glover')	['chicken sandwich', 'chocolate milkshake']

PROBLEM 4 : (*The Contest (44 points)*)

Consider the following programming contest. The contest has several teams participating, and has ten programming problems each team can work on and submit for evaluation. The ten programming problems are identified by the ten letters 'A', 'B', through 'J'. When a programming problem is submitted, the time since the start of the contest is noted in minutes. The program is then evaluated and receives either 'correct', or 'reject' if the code does not work correctly.

A data file with information from a contest has one line per program submission. Each line in the data file is in the following format. First is the name of a team, followed by a colon, followed by a programming contest problem identified by one of ten letters, followed by a colon, followed by the time in minutes of the submission, followed by a colon, and then followed by the word correct or reject.

An example of the data file might be the file `contestdata` below. The first line indicates UNC submitted a program for problem A 20 minutes after the start and it was rejected. The second line indicates Duke submitted a program for problem A at 26 minutes and it was judged correct.

```
UNC:A:20:reject
Duke:A:26:correct
UNC:A:33:reject
ECU:A:34:correct
Elon:A:34:correct
USC:G:44:reject
UNC:A:45:correct
NCSU:B:60:reject
USC:C:72:reject
Duke:E:82:reject
USC:C:90:correct
UNC:B:98:reject
NCSU:B:103:correct
NCSU:A:115:correct
USC:A:116:correct
```

```
ECU:F:202:reject
Duke:D:200:correct
Duke:E:210:correct
UNC:B:212:reject
USC:G:220:reject
NCSU:D:222:correct
Elon:H:225:correct
NCSU:H:230:reject
```

Assume the function `fileToList` has already been written that takes a file in this format and assigns the variable `data` a list of lists of four strings from each line in the file representing the team, the problem letter, the time and the evaluation of the program.

For example, after running `data = fileToList(contestdata)` on the file `contestdata` from the previous page, then `data` would be a list of lists (not all parts shown):

```
data = [ ['UNC', 'A', '20', 'reject'],
['Duke', 'A', '26', 'correct'],
['UNC', 'A', '33', 'reject'],
  ... rest not shown ..
]
```

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 `listOfSchools` which has one parameter, `data`, that is a nonempty list of lists of four strings in the format mentioned earlier. This function returns a sorted list of the unique schools that submitted a program for any problem, regardless of whether it was correct or not.

For example, if data is the list of lists mentioned earlier that was created from the file `contestdata`, then `listOfSchools(data)` would return the list ['Duke', 'ECU', 'Elon', 'NCSU', 'UNC', 'USC']. Each of these schools submitted at least one problem, regardless of whether it was correct or not.

```
def listOfSchools(data):
```

B. (7 pts) Write the function `problemsAttempted` which has one parameter, `data`, that is a nonempty list of lists of four strings in the format mentioned earlier. This function returns a list of the problems attempted, regardless of whether they are correct or not.

For example, if data is the list of lists mentioned earlier that was created from the file `contestdata`, then `problemsAttempted(data)` would return the list ['A', 'C', 'B', 'E', 'D', 'G', 'F', 'H']. Note the order of the problems in the list does not matter.

```
def problemsAttempted(data):
```

C. (7 pts) Write the function `problemsNotAttempted` which has two parameters, `problems`, which is a list of all possible problems ['A', 'B', ... 'J'], and `data`, the list of lists created from the data file.

This function returns a list of the problems that were not attempted by any team.

For example, if data is the list of lists mentioned earlier that was created from the file `contestdata`, then `problemsNotAttempted(problems, data)` would return the list ['I', 'J']. (Note the order of the problems in the list does not matter.)

```
def problemsNotAttempted(problems, data):
```

D. (8 pts) Write the function `dictProblemsToSchoolsSolved` which has one parameter, `data`, that is a nonempty list of lists of four strings in the format mentioned earlier. This function returns a dictionary mapping letters of problems to the list of schools that solved that problem.

For example, if data is the list of lists mentioned earlier that was created from the file `contestdata`, then `dictProblemsToSchoolsSolved` would return a dictionary with several entries, one would be 'D' mapped to the list ['Duke', 'NCSU'], as they are the only schools that got problem 'D' correct.

```
def dictProblemsToSchoolsSolved(data):
```

E. (7 pts) Write the function `dictSchoolsToNumSubmissions` which has one parameter, `data`, that is a nonempty list of lists of four strings in the format mentioned earlier. This function returns a dictionary mapping schools to the number of submissions the school had. Note that a school may submit the same problem multiple times.

For example, if data is the list of lists mentioned earlier that was created from the file `contestdata`, then `dictSchoolsToNumSubmissions` would return a dictionary with several entries, one would be 'Duke' mapped to 4, since Duke submitted A once, D once and E twice, for a total of 4 submissions.

```
def dictSchoolsToNumSubmissions(data):
```

F. (8 pts) Write the function `easiestProblem` which has one parameter, `data`, that is a nonempty list of lists of four strings in the format mentioned earlier. This function returns a tuple of two items, the first is a string for the problem that was solved by the most number of schools, and second is a list of the schools in alphabetical order that solved that problem. If there is a tie, then pick any of those tieing to return.

For example, if data is the list of lists mentioned earlier that was created from the file `contestdata`, then `easiestProblem` would return the tuple ('A', ['Duke', 'ECU', 'Elon', 'UNC', 'NCSU', 'USC']), since problem A had the most number of schools get it correct.

```
def easiestProblem(data):
```