(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

```
letters = ['b', 'e', 'e', 'w']
letters.pop()
print letters
letters.append('h')
print letters
```

```
letters = ['a', 'j', 'k', 's', 'a', \
          'c', 'k', 'b', 'k']
aset = set(letters)
aset.add('p')
alist = list(aset)
alist.sort()
print alist
```

```
lista = [1, 3, 11, 8, 15]
seta = set(lista)
setb = set([5, 10, 15, 2, 1 ])
print seta.intersection(setb)
print seta.union(setb)
print seta.difference(setb)
```

```
dicta = {'C':10, 'D':25, 'U':12, 'F':32}
print dicta['F']
dicta['C'] = 18
dicta['H'] = 25
print dicta.keys()
items = set(dicta.values())
print items
```

```
dicta = {'C':10, 'D':25, 'U':12, 'F':32, 'M':12}
dictb = {}
lista = sorted(dicta.keys())
for key in lista:
    value = dicta[key]
    dictb[value] = key
keylist = dictb.keys()
```

```
keylist.sort()
print keylist
```

This page intentionally blank.

PROBLEM 2 : (*List Comprehensions, eh? (12 points)*)

A. (6 pts) Consider the following code.

```
nums = [5, 7, 2, 3, 9]

lista = [x*2 for x in nums if x > 5]
print lista

words = ['basketball', 'football', 'tennis', 'swimming', 'baseball', 'racquetball']

listb = [w for w in words if w[0] == 's' or w[-1] == 's']
print listb

listc = [w[w.find('as')+2:] for w in words if w.find('as')> 0]
print listc
```

What is the resulting output of this code?

Consider the list `shapes` for the next two problems. Your code should also work if it contains other words.

shapes = ['circle', 'triangle', 'star', 'square', 'diamond']

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 one line of code that includes a list comprehension to assign to the variable `firstlast` a list of **sorted** strings that represent **the first and last letter of each of the words** in the list `shapes`.

For example, using the list `shapes` from above, then after executing the list comprehension, then `firstlast` would be the list ['ce', 'dd', 'se', 'sr', 'te']. Note this list is sorted.

Write the list comprehension below.

firstlast =

C. (3 pts) Write one line of code that includes a list comprehension to assign to the variable `whereIsA` a list of tuples formed from the words in the list `shapes`. The first item in each tuple is a word from the list and the second item is the first position of the letter 'a' in the word. **Tuples are only formed from words that have the letter a.**

For example, using the list `shapes` from above, then after executing the list comprehension, then `whereIsA` would be the list (note circle does not appear and the order of the tuples is the same order the words appear in shapes): [('triangle', 3), ('star', 2), ('square', 3), ('diamond', 2)]

Write the list comprehension below.

whereIsA =

PROBLEM 3 : (*Flying high (15 points)*)

A. (7 pts) Consider the following data file on airline flights for a particular day. The format of each line in the file has the airline as one word, followed by a blank, followed by the flight number, followed by a colon, followed by a city, followed by a colon, followed by the number of passengers.

An example of the data file might be:

```
AA 634:Chicago:247
AA 122:New York:150
Delta 274:New York:226
Delta 5273:Atlanta:88
Delta 1432:Atlanta:67
AA 34:San Francisco:252
AA 315:Memphis:78
AA 457:Atlanta:232
AA 517:Atlanta:34
Delta 1362:Atlanta:167
```

Write the function `fileToList` that has one parameter `datafile` which represents a file that is already open and ready for reading. This function returns a list of lists of items in which each inner list is three strings and an integer. The first item is a string of the airline name, the second item is a string of the flight number, the third item is a string of the city, and the fourth item is an **integer** of the number of passengers on that flight.

For example, the call `fileToList(flightsApril2)` where flightsApril2 is the file above that is open and ready to read would return the list (not all parts shown):

```
[ ['AA', '634', 'Chicago', 247], ['AA', '122', 'New York', 150],
['Delta', '274', 'New York', 226], ['Delta', '5273', 'Atlanta', 88],  ...
]
```

Complete the function fileToList below.

```
def fileToList(datafile):     # Assume datafile open for reading
```

B. (8 pts) Write the function `largeFlights` that has two parameters, `datafile` and `size`, where datafile is the datafile from Part A that is open and ready to read.

This function returns a list of strings, where each string has information about a flight that has `size` or more passengers. The format of each string is "airline-flightnumber-city".

For example, assume *flightsApril2* is the file of data from Part A. The two examples show the result of calling largeFlights with the data file from part A and with different size numbers. There are three flights with 230 or more passengers and one flight with 250 or more passengers.

call	**returns**
largeFlights(flightsApril2, 230)	['AA-634-Chicago', 'AA-34-San Francisco', 'AA-457-Atlanta']
largeFlights(flightsApril2, 250)	['AA-34-San Francisco']

```
def largeFlights(datafile, size):
    datalist = fileToList(datafile)  # assume fileToList works correctly
```

PROBLEM 4 : (*Basketball season (43 points)*)

Consider the new college Brodhead University (BU) that has started a basketball team.

A data file with information about BU's players and its opponents is in the following format. Each line represents a player's score against an opponent and has the name of an opponent(one word), followed by a colon, followed by the player's name (one word), followed by a colon, followed by the number of points scored by that player against that opponent, followed by a colon, followed by "won" or "lost" for the outcome of that game. **Assume BU only plays each opponent once and only players who scored in a game are in the file**

An example of the data file might be the file `scoresfile` below. The first line indicates player Bolton scored two points in a losing game against Duke. The second line indicates player Stone scored 12 points in a winning game against NCSU.

```
Duke:Bolton:2:lost
NCSU:Stone:12:won
Duke:Kreitz:3:lost
Duke:Pura:6:lost
GT:Dolgin:4:lost
WFU:Laveman:20:won
ECU:Parlin:15:won
UNC:Stone:17:won
UNC:Dolgin:12:won
UNC:Kreitz:5:won
```

```
Duke:Stone:16:lost
Duke:Laveman:13:lost
NCSU:Kreitz:8:won
NCSU:Dolgin:18:won
NCSU:Parlin:13:won
GT:Bolton:7:lost
GT:Stone:9:lost
WFU:Parlin:14:won
ECU:Laveman:16:won
ECU:Pura:15:won
```

Assume the function `fileToList` has already been written that takes a file in this format and assigns the variable `datalist` to a list of lists of four strings from each line in the file representing the opponent, the player, the player's score, and whether they won or lost.

For example, after running `datalist = fileToList(scoresfile)` on the file `scoresfile` from above, then `datalist` would be a list of lists (not all parts shown):

```
datalist = [ ['Duke', 'Bolton', '2', 'lost'], ['NCSU', 'Stone', '12', 'won'],
['Duke', 'Kreitz', '3', 'lost'], ['Duke', 'Pura', '6', 'lost'],
  ... 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 `listOfOpponents` which has one parameter, `datalist`, 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 BU played against**.

For example, if datalist is the list of lists mentioned earlier that was created from the file mentioned earlier, then `listOfOpponents(datalist)` would return the list ['Duke', 'ECU', 'GT', 'NCSU', 'UNC', 'WFU']. Note this list is sorted.

```
def listOfOpponents(datalist):
```

B. (7 pts) Write the function `playersScoredLosingTeam` which has one parameter, `datalist`, that is a nonempty list of lists of four strings in the format mentioned earlier. This function returns a **unique** list of the BU players who scored eight or fewer points in any game BU lost.

For example, if data is the list of lists mentioned earlier that was created from the file mentioned earlier, then `playersScoredLosingTeam(datalist)` would return the list ['Pura', 'Dolgin', 'Bolton', 'Kreitz'] Note the order of the names in the list does not matter.

```
def playersScoredLosingTeam(datalist):
```

C. (7 pts) Write the function `teamsNotPlayed` which has two parameters, `league`, which is a list of all the teams in the league BU is in. For example, it might be league = ["Duke", "VT", "UV", "UNCW", "UNC", "WFU", "NCSU", "ECU", "USC", "GT"], and `datalist`, the list of lists created from the file.

This function returns a list of the teams from the league that BU did not play against.

For example, if datalist is the list of lists mentioned earlier and if BU were the league list above, then `teamsNotPlayed(datalist, league)` would return the list ['UNCW', 'USC', 'UV', 'VT']. (Note the order of the teams in the list does not matter.)

```
def teamsNotPlayed(datalist, league):
```

D. (7 pts) Write the function `dictPlayerToGamesPlayedIn` which has one parameter, `datalist`, that is a nonempty list of lists of four strings in the format mentioned earlier. This function returns a dictionary mapping players to the number of games they have played in.

For example, if datalist is the list of lists mentioned earlier, then `dictPlayerToGamesPlayedIn(datalist)` would return a dictionary with several entries, one would be 'Stone' mapped to 4, another would be "Parlin" mapped to 3.

```
def dictPlayerToGamesPlayedIn(datalist):
```

E. (8 pts) Write the function `dictPlayersToOpponentsBeat` which has one parameter, `datalist`, that is a nonempty list of lists of four strings in the format mentioned earlier. This function returns a dictionary mapping players to the list of opponents they beat.

For example, if datalist is the list of lists mentioned earlier, then `dictPlayersToOpponentsBeat(datalist)` would return a dictionary with several entries, one would be 'Stone' mapped to ['NCSU', 'UNC']. Another would be 'Kreitz' mapped to ['UNC', 'NCSU'].

```
def dictPlayersToOpponentsBeat(datalist):
```

F. (7 pts) Write the function `playerOnMostGamesWon` which has one parameter, `datalist`, that is a nonempty list of lists of four strings in the format mentioned earlier. This function returns the name of the player who played in the most games that were won. If there is a tie, then return any of those names.

For example, if datalist is the list of lists mentioned earlier, then `playerOnMostGamesWon(datalist)` would return Parlin. The code has been started below for you.

```
def playerOnMostGamesWon(datalist):
    d = dictPlayersToOpponentsBeat(datalist)  # Assume works correctly
```

This page intentionally blank. This page intentionally blank.