PROBLEM 1:  (What is the output? (16 pts))

For the following code, write the output to the right of each print statement.

```python
lista = ['car', 'ant', 'bat', 'ant']
print(sorted(lista))
#-------------------------------------------------
lista = ['ant', 'bat', 'ant']
listb = [(w, lista.count(w)) for w in set(lista)]
print(sorted(listb))
#-------------------------------------------------
lista = ['flower','red','bug','bark']
listb = sorted(lista, reverse=True)
print(listb)
#-------------------------------------------------
lista = ['red','bug','bark']
listb = sorted(lista, key=len)
print(listb)
#-------------------------------------------------
lista = [(7,3),(8,4),(5,2)]
listb = sorted(lista, key=min)
print(listb)
#-------------------------------------------------
lista = [(‘H’,5),('E',8),('M',2)]
listb = sorted(lista, key=lambda x:x[1])
print(listb)
#-------------------------------------------------
d = {‘M’: [8, 9], ‘C’:[7,12], ’Y’:[4,10]}
ans = sorted(d.keys())
print(ans)
ans = sorted(d.items(), key=lambda x:x[1][1])
print(ans[-1])
```

OUTPUT:

```python
['bat', 'ant', 'car']
[(‘ant’, 2), (‘bat’, 1), (‘ant’, 1)]
['flower', 'bug', 'bark', 'red']
['red', 'bug', 'bark']
[(7, 3), (5, 2), (8, 4)]
[(‘M’, 2), (‘E’, 8), (‘H’, 5)]
[‘M’, ‘C’]
```
PROBLEM 1: (What is the output? (16 pts))

For the following code, write the output to the right of each print statement.

```python
lista = ['mat', 'top', 'mat', 'bee']
print(sorted(lista))

#-------------------------------------------------
lista = ['mat', 'car', 'mat']
listb = [(w, lista.count(w)) for w in set(lista)]
print(sorted(listb))
#-------------------------------------------------
lista = ['leaf', 'top', 'acorn', 'hill']
listb = sorted(lista, reverse=True)
print(listb)
#-------------------------------------------------
lista = ['tree', 'car', 'bug']
listb = sorted(lista, key=len)
print(listb)
#-------------------------------------------------
lista = [(6,2),(7,3),(4,1)]
listb = sorted(lista, key=min)
print(listb)
#-------------------------------------------------
lista = [('J',4),('F',7),('N',1)]
listb = sorted(lista, key=lambda x:x[1])
print(listb)
#-------------------------------------------------
d = {'N': [7, 8], 'D':[6,11], 'Z':[3,9]}
ans = sorted(d.keys())
print(ans)

ans = sorted(d.items(), key=lambda x:x[1][1])
print(ans[-1])
```

OUTPUT:
Complete the following functions. This problem has three parts.

PART A (6 pts)
Write the function named `update` that has five parameters. The first parameter named `d` is a dictionary, where each key is a string mapped to a list of three integers. The second parameter is a string named `key`. The last three parameters are integers named `val1`, `val2` and `val3`. This function updates and returns the dictionary `d`, updating `key`'s value by adding `val1` to the first value in its list, adding `val2` to the second value in its list, and adding `val3` to the third value in its list.

For example, assume the dictionary `d` is the following:

\[ d = \{ 'A': [0, 1, 2], 'B': [1, 0, 0], 'C': [0, 1, 1], 'F': [2, 2, 2] \} \]

After the call `update(d, 'C', 1, 2, 3)` , the dictionary is below. Note that before the call 'C' was mapped to the list [0, 1, 1] and after the call C’s list is updated and is now [1, 3, 4].

\[ d = \{ 'A': [0, 1, 2], 'B': [1, 0, 0], 'C': [1, 3, 4], 'F': [2, 2, 2] \} \]

Complete the function below.

```python
def update(d, key, val1, val2, val3):
```
PART B (6 pts)

Write the function named `match` that has two string parameters named `word1` and `word2`. Assume these words are the same length, you do not need to verify their length is the same. This function returns a list of strings, each of length 1, in which the $k$th string is the $k$th letter of `word1` if the $k$th letter of `word1` is the same as the $k$th letter of `word2`. Otherwise, the $k$th string is the underscore `_'`. The list returned should be the length of `word1`.

We give several examples of calls to this function.

<table>
<thead>
<tr>
<th>call</th>
<th>returns</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>match(&quot;teach&quot;,&quot;trick&quot;)</code></td>
<td>['t', '_', <em>, 'c', '</em>']</td>
<td>t and c match in same position</td>
</tr>
<tr>
<td><code>match(&quot;teach&quot;,&quot;beach&quot;)</code></td>
<td>['_', 'e', 'a', 'c', 'h']</td>
<td>all but first letter match</td>
</tr>
<tr>
<td><code>match(&quot;cat&quot;,&quot;and&quot;)</code></td>
<td>['<em>', '</em>', '_']</td>
<td>no letters match in same position</td>
</tr>
</tbody>
</table>

Complete the function below.

```python
def match(word1, word2):
```
PART C (6 pts)
Write the function named `someOrder` that has one parameter named `lista` that is a list of
tuples, where each tuple has three integers.
This function returns a list of the tuples sorted in the following way:

1. sorted by the first number in each tuple, in reverse order
2. break ties by sorting by the last number in each tuple
3. after both 1) and 2) break ties by sorting by the middle number in reverse order

For example, the call `someOrder([(0, 3, 4), (2, 3, 4), (2, 1, 4), (3, 1, 4), (0, 5, 3)])` re-
turns the list `[(3, 1, 4), (2, 3, 4), (2, 1, 4), (0, 5, 3), (0, 3, 4)]`.

Complete the function below.

def someOrder(lista):
PART C (6 pts)

Write the function named `someOrder` that has one parameter named `lista` that is a list of tuples, where each tuple is a string followed by two integers. This function returns a list of the tuples sorted in the following way:

1. sorted by the second item in each tuple, in reverse order
2. break ties by sorting by the length of the first item in each tuple
3. after both 1) and 2) break ties by sorting by the last item in each tuple in reverse order

For example, the call

```python
someOrder([('cat',3,2), ('ab',3,1), ('a',1,4), ('dog',3,4), ('j',5,3)])
```

returns the list

```python
[('j',5,3), ('ab',3,1), ('dog',3,4), ('cat',3,2), ('a',1,4)]
```

Complete the function below.

```python
def someOrder(lista):
```
PROBLEM 3:  (Playing Games (35 pts))

This problem is about data related to card games at a tournament. There are several card games that can be played. Each game is played between two players and results in either one player winning or a tie. A player who wins gets 1.0 point, a player who loses gets 0.0 points, and if there is a tie, then both players get 0.5 points.

There are five functions to write in this part. Your functions should work for any valid data, not just the examples shown.

Most of the problems have datalist as one of the parameters. The parameter datalist is a list of lists, with each inner list representing one game played between two players and having the following five items: 1) a string representing player1’s name 2) player1’s score as a float 3) a string representing player2’s name 4) player2’s score as a float and 5) a string of the name of the game the two players’ played. We will assume player’s names are unique.

For example, assume datalist is the lists of lists shown below. The first inner list shows Cristina and Chloe played game C, and Cristina won. Cristina received a score of 1.0 and Chloe received a score of 0.0. The second inner list shows Finn and Erin played game F, and Erin won. Erin received a score of 1.0 and Finn received a score of 0.0. The third inner list shows Brendan and Adam played game E and ended in a tie. Both Brendan and Adam received a score of 0.5.

datalist = [['Cristina', 1.0, 'Chloe', 0.0, 'C'], [
  ['Finn', 0.0, 'Erin', 1.0, 'F'],
  ['Bala', 1.0, 'Wei', 0.0, 'F'],
  ['Adam', 0.5, 'Cristina', 0.5, 'H'],
  ['Tali', 0.0, 'Angel', 1.0, 'A'],
  ['Adam', 0.5, 'Finn', 0.5, 'E'],
  ['Bala', 1.0, 'Wei', 0.0, 'F'],
  ['Sophia', 0.0, 'Saad', 1.0, 'C'],
  ['Chloe', 1.0, 'Adam', 0.0, 'B'],
  ['Sophia', 0.0, 'Brendan', 1.0, 'C'],
  ['Tali', 1.0, 'Sophia', 0.0, 'F'],
  ['Wei', 1.0, 'Saad', 0.0, 'A'],
  ['Brendan', 0.5, 'Bala', 0.5, 'F']
]}

In solving the problems that follow, you may call any of the other functions in this problem. Go to the next page to start Part A of this problem.
Part A (7 pts)
Write the function named gamesPlayed that has one parameter named datalist, which is a list of lists in the format described earlier.

We repeat the format of datalist, a list of lists with each inner list representing one game played between two players and having the following five items: 1) a string representing player1’s name 2) player1’s score as a float 3) a string representing player2’s name 4) player2’s score as a float and 5) a string of the name of the game the two players’ played. We will assume player’s names are unique.

This function returns a sorted list of tuples of pairs, each pair is the name of a game and how many times that game was played. The list is sorted by the number of times games were played, in increasing order. If some games were played the same number of times, those games can be listed in no particular order.

For example, assume datalist is the list of lists shown on the first page of this problem. Calling gamesPlayed(datalist) would return the following sorted list. Note game 'H' was played once, game 'B' was played twice, both games 'E' and 'A' were played three times, and both games 'C' and 'F' were played 5 times.

\[\left[\text{('H', 1), ('B', 2), ('E', 3), ('A', 3), ('C', 5) }, ('F', 5)\right]\]

Complete the function below.

def gamesPlayed(datalist):
Part B (7 pts)

Write the function named `gamesNotPlayed` that has two parameters, The first parameter is named `datalist`, which is a list of lists in the format described earlier, and the second parameter named `allgames` is a list of all possible games to play.

We repeat the format of `datalist`, a list of lists with each inner list representing one game played between two players and having the following five items: 1) a string representing player1’s name 2) player1’s score as a float 3) a string representing player2’s name 4) player2’s score as a float and 5) a string of the name of the game the two players’ played. We will assume player’s names are unique.

This function returns a sorted list of the games not played at this tournament.

If the list of all games is `['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']`, then the call `gamesNotPlayed(datalist, allgames)` would return the sorted list: `['D', 'G']`

Complete the function below.

```python
def gamesNotPlayed(datalist, allgames):
```
Part C (7 pts)

Write the function named `gamesWon` that has one parameter named `datalist`, which is a list of lists in the format described earlier.

We repeat the format of `datalist`, a list of lists with each inner list representing one game played between two players and having the following five items: 1) a string representing player1’s name 2) player1’s score as a float 3) a string representing player2’s name 4) player2’s score as a float and 5) a string of the name of the game the two players’ played. We will assume player’s names are unique.

This function returns a dictionary of each player mapped to the list of games they won. Calling `gamesWon(datalist)` on the `datalist` given would result in returning the following dictionary (Cristina won games C and A, Erin won game F, etc.):

```python
{ 'Cristina': ['C', 'A'], 'Erin': ['F'], 'Bala': ['F'], 'Saad': ['C'], 'Angel': ['A'], 'Wei': ['F', 'A'], 'Chloe': ['B', 'C'], 'Brendan': ['C'], 'Tali': ['F'], 'Kaitlyn': ['B'] }
```

Complete the function below.

```python
def gamesWon(datalist):
```
Part D (7 pts)
Write the function named mostWon that has one parameter named dictwon, which is a
dictionary of each player mapped to a list of the games they have won.
This function returns a sorted list of the games that were won the most. For example,
suppose the dictionary passed in is the following:

dictwon = { 'Cristina': ['C', 'A'], 'Erin': ['F'], 'Bala': ['F'], 'Saad': ['C'],
'Angel': ['A'], 'Wei': ['F', 'A'], 'Chloe': ['B', 'C'], 'Brendan': ['C'],
'Tali': ['F'], 'Kaitlyn': ['B'] }

Then the games won the most are C and F (both won 4 times). The call mostWon(dictwon)
would return the sorted list ['C', 'F'].
Complete the function below. Note that you do not have datalist here, but dictwon instead.

def mostWon(dictwon):

Part E (7 pts)
Write the function named gamesOnlyTied that has one parameter named datalist, which is a list of lists in the format described earlier.

We repeat the format of datalist, a list of lists with each inner list representing one game played between two players and having the following five items: 1) a string representing player1’s name 2) player1’s score as a float 3) a string representing player2’s name 4) player2’s score as a float and 5) a string of the name of the game the two players’ played. We will assume player’s names are unique.

This function returns a sorted list of the unique names of games that were played at least once, but no one ever won (they always ended in a tie).

For example, consider the datalist example given at the beginning of this problem. Game ‘E’ was played three times always ending in a tie. Game ‘H’ was played only once and ended in a tie. For all the other games played, each had at least one pair of players with a winner. The call gamesOnlyTied(datalist) would return the sorted list: ['E', 'H'].

Complete the function below.

def gamesOnlyTied(datalist):