

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','N','V']
for (i,e) in enumerate(lista):
    print i,e
lista.append('R')
print lista
lista.insert(2,'B')
lista.pop()
print lista
#-----
listb = ['S','U', 'S', 'D', 'U']
setb = set(listb)
print setb
setb.add('D')
setb.add('P')
print sorted(list(setb))
#-----
seta = set([4,8,2,1,8])
setb = set([8,4,9,1])
print seta.intersection(setb)
print seta.union(setb)
print seta.difference(setb)
#-----
dicta = {'M':9, 'B':3, 'K':5, 'F':7}
print dicta['B']
dicta['Y'] = 10
dicta['K'] = 9
print sorted(dicta.keys())
print sorted(dicta.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.

```

animals = ['rhino', 'cat', 'dog', 'rabbit', 'reindeer', 'camel', 'robin']
d = {}
for w in animals:
    if w[0] not in d:

```

```
        d[w[0]] = 1
    else:
        d[w[0]] += 1

print d.keys()
print d.values()
print max(d.values())
print max(d.keys())
```

What is the output?

PROBLEM 2 : (*Short Code/Answer (20 points)*)

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

For parts that involve writing code, your code should also work if the list `words` was modified to contain different words.

```
words = ['star', 'yes', 'orange', 'money', 'sassy', 'hair']
```

A. (3 pts) Consider the following code.

```
temp = sorted([w for w in words if w[0]=='y' or w[-1] == 'y'])
```

What is the value of `temp` after this line executes?

B. (3 pts) Consider the following code.

```
temp = [{"s"*w.count('s'), w} for w in words if 's' in w]
```

What is the value of `temp` after this line executes?

C. (4 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` that are larger than the first word in `words`. The resulting list should have the words in the same order as the original list.

For example, if `words` was the list above, then after executing the list comprehension, then `temp` would be the list `['orange', 'money', 'sassy']`.

Write the list comprehension below.

```
temp =
```

```
words = ['star', 'yes', 'orange', 'money', 'sassy', 'hair']
```

D. (5 pts) Write code (could be more than one line) that **includes a list comprehension** to assign to the variable `temp` a list of words from the list `words` such that all the words that start with 's' are first and in sorted order, and the remaining words follow, also in sorted order.

For example, if `words` was the list above, then after executing the list comprehension, then `temp` would be the list `['sassy', 'star', 'hair', 'money', 'orange', 'yes']`

Write the code with at least one list comprehension below.

```
temp =
```

E. (5 pts) Write a function named `modify` that has one parameter `namelist` that is a list of strings. Each string is one or more words separated by a blank. This function returns a list of modified strings from `namelist` such that if a string was in `namelist` and had exactly two words, then the words are swapped and this new string is part of the new list returned.

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

```
names = ['hello kitty', 'darth vader', 'prince', 'jolly rodger', 'long john silver']
```

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

```
['kitty hello', 'vader darth', 'rodger jolly']
```

 Note that only those names with two words are modified and included, and appear in the same order they were in the original list.

```
def modify(namelist):
```

PROBLEM 3 : (Time to Register for Courses (15 points))

Consider the following data file on students registering for courses for the following semester. The format of each line in the file has the course (one word), followed by a slash ('/'), followed by the last name of the student, followed by a slash ('/'), followed by the number of seats for the course, followed by a slash ('/'), followed by the position number the student is in for registering for this course (if the number is "5", they are the 5th person to register for this course).

An example of the data file is shown below. For example, in the first line, Hester has registered for CompSci 101. There are 200 seats and Hester is the 207th person to sign up, meaning Hester is 7th on the waitlist. In the second line. Morgan has registered for Econ101. There are 150 seats and Morgan is the 33rd person to register, so Morgan is registered for the course.

```
CompSci101/Hester/200/207
Econ101/Morgan/150/33
```

```
Math230/Hester/28/17
Stats111/Huang/60/81
Bio158/Ching/25/8
Russian101/Beck/25/29
Econ101/Tobin/150/155
Sociol160/Farrell/18/20
CompSci101/Beck/200/213
Econ101/Ching/150/151
Bio158/Huang/25/25
Bio158/Tobin/25/30
Bio158/Taylor/25/29
Math230/Huang/28/32
Stats111/Beck/60/64
Sociol160/Hester/18/21
```

A. (7 pts) Write the function `processData` 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 is four strings. The first item is a string of the course, the second item is a string of the name of the person, the third item is a string of the number of seats in the course, and the fourth item is a string of the position number the student is in for registering for this course (if the number is "5", they are the 5th person to register for this course).

For example, the line `data = processData("registration.txt")` where "registration.txt" is the file on the previous page would result in `data` having the value below.

```
data = [ ['CompSci101', 'Hester', '200', '207'],
['Econ101', 'Morgan', '150', '33'],
['Math230', 'Hester', '28', '17'],
['Stats111', 'Huang', '60', '81'],
['Bio158', 'Ching', '25', '8'],
['Russian101', 'Beck', '25', '29'],
['Econ101', 'Tobin', '150', '155'],
['Sociol160', 'Farrell', '18', '20'],
['CompSci101', 'Beck', '200', '213'],
['Econ101', 'Ching', '150', '151'],
['Bio158', 'Huang', '25', '25'],
['Bio158', 'Tobin', '25', '30'],
['Bio158', 'Taylor', '25', '29'],
['Math230', 'Huang', '28', '32'],
['Stats111', 'Beck', '60', '64'],
['Sociol160', 'Hester', '18', '21'] ]
```

Complete the function `processData` below.

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

B. (8 pts) Write the function `peopleWaitlisted` that has two String parameters, `filename` and `coursename`, where `filename` is the name of a file that is in the format from Part A.

This function returns a list of tuples of information about each person waitlisted in `coursename`. Each tuple has the position on the waitlist first, followed by the persons name.

For example, assume `filename` is the file on the first page of this problem. The two examples below show the result of calling `peopleWaitlisted` with this filename and a course name. For example, Hester was the 207th person to register for CompSci 101, which had a seat limit of 200. Hester is thus 7th on the waitlist for CompSci 101. Beck is the 13th person waitlisted for CompSci 101 Note only those waitlisted for a course are shown.

call	returns
<code>peopleWaitlisted(filename,"CompSci101")</code>	<code>[(7, 'Hester'), (13, 'Beck')]</code>
<code>peopleWaitlisted(filename,"Bio158")</code>	<code>[(5, 'Tobin'), (4, 'Taylor')]</code>

```
def peopleWaitlisted(filename, coursename):  
    data = processData(filename)
```

PROBLEM 4 : (*Playing cards (39 points)*)

Suppose you have designed a new deck of cards and you are selling the decks on ebay.

Data from your deck sales is stored in a list of lists, where each list represents a purchase from a person of one or more decks of cards. This list of lists has four strings: a state abbreviation for the state the deck was sold in, a name of the person who bought the deck, the person's age, and the number of decks sold for this purchase.

For example, suppose `datalist` is the list below. The first list in `datalist` represents that someone named Koch in NC who is age 14 bought 1 deck of cards. The second list in `datalist` indicates that someone named Mitchell from CA who is age 24 bought 2 decks.

```
datalist = [ ['NC', 'Koch', '14', '1'],  
            ['CA', 'Mitchell', '24', '2'],  
            ['SC', 'Guo', '42', '5'],  
            ['KY', 'Jin', '55', '3'],  
            ['MT', 'Lopez', '31', '1'],  
            ['CA', 'Tang', '24', '2'],  
            ['KY', 'Noor', '28', '10'],  
            ['SC', 'Guo', '42', '8'],
```

```
['NC', 'Koch', '14', '1'],
['CA', 'Mitchell', '24', '4'],
['CA', 'Dacon', '55', '12'],
['SC', 'Guo', '42', '5'],
['MT', 'Lopez', '31', '5'],
['CA', 'Tang', '24', '1'],
['SC', 'Guo', '42', '8'],
['KY', 'Huh', '24', '4'] ]
```

Note that the same person may have bought decks more than once and thus, may appear more than once in the list.

A. (7 pts) Write the function `UniqueStates` 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 states of people who bought card decks.

For example, if `datalist` is the example list of lists mentioned earlier then `UniqueStates(datalist)` would return the list `['CA', 'KY', 'MT', 'NC', 'SC']`. Note this list is sorted.

```
def UniqueStates(datalist):
```

B. (7 pts) Write the function `UniqueNamesByState` which has two parameters, `datalist`, that is a nonempty list of lists of four strings in the format mentioned earlier, and a `String state`. This function returns a **sorted unique** list of the names that bought at least one deck in the given state.

For example, if `datalist` is the list of lists mentioned earlier that was created from the file mentioned earlier, then `UniqueNamesByState(datalist, 'NC')` would return the list `['Koch']`, and `UniqueNamesByState(datalist, 'CA')` would return the list `['Dacon', 'Mitchell', 'Tang']` Note the names are in sorted order.

```
def UniqueNamesByState(datalist, state):
```

C. (7 pts) Write the function `PeopleNotTargeted` which has two parameters, `datalist`, that is a nonempty list of lists of four strings in the format mentioned earlier, and a list named `targets` that is a list of people that you specifically reached out to to encourage them to purchase decks. This function returns a **sorted** list of people who bought cards but were not targetted (you did not reach out to them).

For example, if `datalist` is the list of lists mentioned earlier and if `targets` is the list `['Jin', 'Koch', 'Guo', 'Dacon', 'Mitchell']` then the call `PeopleNotTargeted(datalist, targets)` would return the **sorted** list `['Huh', 'Lopez', 'Noor', 'Tang']`.

```
def PeopleNotTargeted(datalist, targets):
```

D. (8 pts) Write the function `MostCardsByOnePerson` which has one parameter, `datalist`, that is a nonempty list of lists of four strings in the format mentioned earlier. This function calculates the total number of cards bought by each person, and returns the largest such number. **You must build a dictionary as part of solving this problem.**

For example, if `datalist` is the list of lists mentioned earlier, then `MostCardsByOnePerson(datalist)` would build a dictionary with several entries, one would be 'Dacon' mapped to 12, another would be 'Guo' mapped to 26 (Note 'Guo' bought four times with a total of 26 decks), and there would be others with smaller values. This function would return 26 since it is the largest number bought by any person. Assume the names are unique, though a person could make more than one purchase so could appear multiple times in the data.

```
def MostCardsByOnePerson(datalist):
```

E. (10 pts) Write the function `StateWithMostPeople` which has one parameter, `datalist`, that is a nonempty list of lists of four strings in the format mentioned earlier. This function calculates for each state, the number of different people who bought cards, and returns the state that has the largest number of such people. If there is a tie, return any of those with the most.

For example, if `datalist` is the list of lists mentioned earlier, then `StateWithMostPeople` would determine that SC had one person buy cards, named 'Guo', (note this person bought decks four different times, but it is the same person), CA had 3 people, KY had 3 people and there were a few more, but all smaller than 3. So either 'CA' or 'KY' should be returned since they both had the most.

```
def StateWithMostPeople(datalist):
```