PROBLEM 1 : (***What is the output? (14 points)***)

What is the output of the following code segment? Write the output below. Note that there is only output for the print statements. When printing a set, the elements in the set can be printed in any order.

```
spices = ['sage', 'cumin', 'basil', 'sage', 'cloves', 'sage', 'salt', 'cumin']
a = set(spices)
b = list(a)
b.sort(reverse=True)
print b

b.pop()
b.pop()
print b

lista = ['sage', 'salt', 'cumin']
listb = ['basil', 'salt', 'cloves', 'basil']
seta = set(lista)
setb = set(listb)
print seta | setb
print seta & setb
print setb - seta

dict = {'flower':['red', 'blue'], 'tree':['brown'], 'frog':['red', 'green', 'brown']}
print dict.keys()

colors = set([])
for w in dict:
    item = dict.get(w)
    colors.add(item[0])
print colors

OUTPUT:
```

PROBLEM 2 : (***Retake: What is the output? (8 points)***)

What is the output of the following code segment? Write the output below. Note that there is only output for the print statements.

PROBLEM 4 : (*Top-level Domain (12 points)*)

A. (6 pts) The *domain name* of a URL is the first part of the URL up to but not including the "/". The *top-level domain* part of the domain name is the last part of it. For example in the first URL below, the domain name is "calendar.duke.edu" and the top-level domain is "edu".

Write the function `topLevelDomain` that has one string parameter `url` representing a URL. This function returns a string that is the top-level domain.

call	**returns**
topLevelDomain("calendar.duke.edu/events")	'edu'
topLevelDomain("amazon.com")	'com'
topLevelDomain("www.cs.duke.edu/courses/cps006/fall11")	'edu'
topLevelDomain("www.amazon.de/gp/bestsellers/books/ref=svb2")	'de'

```
def topLevelDomain(url):
    '''
    url is a string representing a url. Parts of the url may be
    separated by 0 or more '/'s, the first part will have at
    least one '.' in it.
    returns the last word before the first '/' or the last word
    if there is no '/'
    '''
```

B. (6 pts) Write the function `domainsVisited` that has one parameter `file`, which is a file that is open and ready to read. The file contains urls that someone has visited, one url per line. Return a list of the unique top-level domain names from the file, in alphabetical order.

For example, if the file named `rodgerUrls` contained the urls below and was open and ready to read, then the call `domainsVisited(rodgerUrls)` would return the list `['com', 'de', 'edu', 'org']`.

```
calendar.duke.edu/events
amazon.com/books
amazon.com
www.cs.duke.edu/courses/cps006/fall11
www.jflap.org/tutorial/
www.csc.ncsu.edu/about_us.php
maps.google.com
www.amazon.de/gp/bestsellers/books/ref=sv_b_2
```

Complete the function below. In writing this function you must call the function `topLevelDomain` that you wrote in part A.

```
def domainsVisited(file):
# given 'file' containing urls, in which file is open and ready to read,
# return a sorted list of the unique top-level domain names.
```

PROBLEM 5 : (*A picture is worth a thousand words (16 points)*)

A. (7 pts) Consider the following data file in which each line in the file has the name of a museum (one word), the name of a painter (one word) and then the name of a painting at that museum (one or more words). An example of the data file might be:

```
MET Rubens A forest at dawn with a deer hunt
MET Rubens Venus and Adonis
MET Kensett A foggy sky
MET Bargue A footman sleeping
Louvre Rembrandt Albert Cuyper
MET Picasso Reading at a table
MET Picasso Head of a woman
Louvre daVinci The Mona Lisa
Louvre Rembrandt Supper at Emmaus
Louvre Cezanne Bathers
MET Rembrandt Old woman cutting her nails
NCMA Rembrandt Young man with a sword
```

Write the function `fileToList` that has one parameter `file` 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 name of the museum, the second string is the name of a painter and the third string is the name of a painting by that painter that is at that museum.

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

```
[['MET', 'Rubens', 'A forest at dawn with a deer hunt'],
 ['MET', 'Rubens', 'Venus and Adonis'],
 ['MET', 'Kensett', "A foggy sky'],
  ... rest not shown ..
]
```

Complete the function fileToList on the next page.

```
def fileToList(file):
# 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 museum, artist, painting
```

B. (9 pts) Write the function `artistInMostMuseums` that has one parameter `data`, where data is a list of lists of strings in the format from part A. Each inner list is a list of three strings: the museum, the artist and the name of the painting of the artist that is at that museum. This function returns the name of the artist that has paintings in the most number of distinct museums. Note that it does not matter how many paintings an artist has in a museum, just how many different museums the artist has a painting in.

For example, assume the file at the beginning of this problem has been turned into a list of lists called paintlist. Then the call `artistInMostMuseums(paintlist)` would return "Rembrandt" as it is the only artist that has paintings in 3 museums. The other artists have paintings in 1 or 2 museums. If there is a tie, then return any of the artists that reach the maximum number of museums with paintings.

```
def artistInMostMuseums(data):
# return the name of the artist that has paintings in the most number
# of museums.  In case of a tie return one such artist.
```

PROBLEM 6 : (*Popular fish (14 points)*)

Write the function `popularFish` that has one parameter `data` that is a list of strings, and a second integer parameter `num`. Each string in the list `data` is in the following format: the name of a fish tank, followed by ":", followed by the name of one or more types of fish separated by ":"s. This function returns a list of fish in alphabetical order that have more than "num" fish over all the fish tanks.

For example consider the following list.

```
fishdata = ["tank1:angel:angel:german ram:german ram",
            "tank2:catfish:catfish:algaeeater:angel:killifish:killifish:discus",
            "tank3:killifish:killifish:killifish:catfish:catfish:angel:angel:angel",
            "show tank:killifish:killifish:killifish:killifish:killifish:algaeeater"]
```

The call `popularFish(fishdata,4)` returns ['angel', 'killifish'], since both of them have more than 4 fish total over all the fish tanks. Note the fish are sorted in alphabetical order.

```
def popularFish(data, num):
```

PROBLEM 3 : (*It's a mystery (12 points)*)

Consider the following `mystery` function with one parameter `animals` which is a list of strings.

```
def mystery(animals):                              # line 1
    ''' animals is a list of strings '''
    x = [ ]                                        # line 2
    for w in animals:                              # line 3
        x.append(len(w))                           # line 4
    amount = max(x)                                # line 5
    y = [w for w in animals if len(w) == amount]   # line 6
    return y[0]                                    # line 7
```

A. (4 pts) Consider making the call `mystery(animals)` with the value of `animals` below. Answer the following questions about tracing what happens with this call

```
animals = ['cat', 'mouse', 'snake', 'chicken', 'fish']
```

A1. What is the value of `x` after line 5 executes?

A2. What is the value of `amount` after line 5 executes?

A3. What is the value of `y` after line 6 executes?

A4. What value is returned from the call `mystery(animals)`?

B. (8 pts) Consider making the call `mystery(zoo)` with the value of `zoo` below. Answer the following questions about tracing what happens with this call

```
zoo = ['lion', 'rhino','bear','zebra']
```

B1. What value is returned from the call `mystery(zoo)`?

B2. Explain in words what `mystery` does.

B3. Rewrite lines 2-5 as one line that includes a list comprehension

B4. In the original code, if line 7 was changed to `return y[-1]`, explain in words what `mystery` would now do.

PROBLEM 1 : (*20 points*)

These questions ask you to write code that references the list `words` below. The Python code you write should work with any values stored in the list `words`. *You can write one line of code or many for each of the tasks below.*

```
words = ["eagle", "underground", "uncopyrightable", "go", "angry", "bookkeepers", \
         "hungry", "america", "screeches", "aglet"]
```

Part A (3 points)

Write code to store in the variable `middles`, a list of strings, the middle letter of each of the 5-letter words in the order they appear in the list. In the example this would be `["g", "g", "l"]`.

Part B (5 points)

Write code to store in the variable `sameEnds`, a list of strings, those words that start and end with the same letter in sorted order from the longest to the shortest. In the example this would be `["screeches", "america", "eagle"]`.

Part C (6 points)

Write code to store in the variable `unique`, a list of strings, those words that do not contain repeated letters in sorted order alphabetically. In this example this would be `["aglet", "angry", "go", "hungry", "uncopyrightable"]`

Part D (6 points)

Given the additional variable `onebigword` the combines all the words in the list into one single string:

```
>>> onebigword = "".join(words)
"eagleundergrounduncopyrightablegoangrybookkeepershungryamericascreechesaglet"
```

Write code to store in the variable `mostCommon`, a string, the most common letter in all the words. In this example this would be `"e"`.

PROBLEM 2 : (*12 points*)

The following problem asks you to choose a Python sequence (strings, lists/tuples, sets, or dictionaries) that would most help represent the data described.

For each scenario, state the type of sequence you feel best represents the data and justify your decision. Note, more than one sequence may make sense, so be sure to clearly explain your reasoning. If you suggest a dictionary, clearly state the types and purpose of the keys and values. If you suggest a sequence of sequences (e.g., a list of lists or set of tuples), clearly describe the type and purpose of the sub-sequences.

Consider the problem of managing a major airline corporation, organizing pilots and planes, tracking passenger information, and getting them to their flights. For the following scenarios, you can assume that pilots, planes, and passengers have unique names (or IDs), and in-flight rows are simply numbered (i.e., you do not need to come up with a complex solution to represent items uniquely, a simple string or number will do).

1. For a departing flight, organize passengers such that they board in a pre-determined order.

2. At any airport, there are different kinds of planes (Boeing 747, 787, etc.). How would you determine which kinds of planes are common across airports?

3. On a plane, organize the passengers in their seats in their rows.

4. At the airport, organize passengers based on their departure time so that those with less time can be given priority at the security checkpoint.

PROBLEM 3 : (*12 points*)

The *contestWinner* APT returns the numeric ID of the winner of the contest whose rules are simple: the winner is the contestant who solves the largest number of tasks. If contestants tie for most tasks solved, the winner is the one who was the first to have all of their tasks solved.

You are given a log of all accepted solutions as a list of int values where the i-th element of the list is the number of the contestant who submitted the i-th accepted solution.

As a reminder, this table illustrates what the function is supposed to return:

ID	Call	Expected Value
#1	`contestWinner([4, 7, 4, 1])`	4
#2	`contestWinner([10, 20, 30, 40, 50])`	10
#3	`contestWinner([123, 123, 456, 456, 456, 123])`	456

Consider the following **correct** solutions. For each implementation, explain what it does and why it works as a solution to the problem. Note, each solution is shorter than the one before, though not necessarily better, so your explanation should focus on why it works, not just describing the code, even though it may not seem to consider a case that a previous solution did.

Note, the lines are numbered for your convenience in referring to them in your explanation, they would not be included when the code is run.

Part A (4 points)

```
    def contestWinner (events):
1.    d = {}
2.    for (idx, event) in enumerate(events):
3.        if event not in d:
4.            d[event] = [ idx ]
5.        else:
6.            d[event] += [ idx ]
7.    most = max([ len(x) for x in d.values() ])
8.    winners = sorted([ (v[-1], k) for (k, v) in d.items() if len(v) == most ])
9.    return winners[0][1]
```

Part B (4 points)

```
    def contestWinner (events):
1.      most = max([ events.count(x) for x in events ])
2.      d = {}
3.      for event in events:
4.          if event not in d:
5.              d[event] = 0
6.          d[event] += 1
7.          if d[event] == most:
8.              return event
```

Part C (4 points)

```
    def contestWinner (events):
1.      events.reverse()
2.      tuples = [ (events.count(name), events.index(name), name) for name in set(events) ]
3.      return max(tuples)[2]
```