These practice problems cover strings, file I/O, lists, and dictionaries in the textbook. The exam will have a similar format to past exams: some code reading (i.e., write the result of the expression, or describe what the code does, or say what the output will be) and some code writing.

Don't forget about http://codingbat.com/python --- there are lots of good practice problems at that site.

Also, the site http://pythontutor.com has some useful visualization capabilities. The visualizer features are helpful for understanding "list aliasing" and other curious things that happen when you pass a list into a function and modify the list from inside the function.

Expression problems

Assume that the following statement have already been executed in the interactive interpreter:

```
a = "radio cure"
b = [ 1, 1, 2, 3, 5 ]
c = { 'i':3, 'a': 7, 'y':5, 'need':2 }
```

For each of the following expressions, evaluate the expression and write the resulting value, or identify the error in the code that would prevent it from running.

1.
```
(len(b) / 2) in c
```

2.
```
c[a.split()[2]] > len(a)
```

3.
```
a[2:-2:min(c.values())]
```

4.
```
len(b) in c.values()
```

5.
```
len(b) in c.keys()
```

6.
```
d = b[::2].append(-13).sort()
print d
```

7.

```
b.index(len(c)+1)+1 in c.values()
```

8.

```
c[a[1]] = a[-4:]
e = c.keys()
e.sort()
print e
```

String problems

1. Given a string S with odd length:
   a. write an expression to print the middle character
   b. write an expression to print the string up to but not including the middle character
   c. write an expression to print from the middle character to the end, not including the middle character (i.e., from one after the middle to the end)

2. Given the string s='abcdefghij', write a string slice expression that will print the following:
   a. 'jihgfedcba'
   b. 'adgj'
   c. 'igeca'

3. Write a function named countsubstr that takes two strings as parameters and returns the number of times that the second string appears in the first string. For example:

- countsubstr('aabbabab','ab') should return 3
- countsubstr('computersarecomputational','comp') should return 2

You cannot use any built-in string methods for this problem (but you can use slicing).

4. Write a function makeAcronym that takes a phrase (a string) as a parameter, and creates and returns a new string that contains the first letter of each word in the phrase, capitalized. For example, makeAcronym('Gnus not UNIX') should return the string 'GNU'. For this problem, you can use any string methods you'd like. For example: upper(), lower(), count(x), find(x), or split().

5. Write a function that takes a string as input and returns the string with all 'a's removed. For example, if you call removeAs('amalgamate'), the function should return the string 'mlgmte'. Your function should remove both lower-case and upper-case A's but should not change any other characters. (You should be able to write the function recursively, too -- though this won't be on exam 3.)

6. Write a function that takes a name in the form of "Doyle, Arthur Conan", and returns the name in the form "Arthur Conan Doyle".

7. Write a function that takes 1 word (a string) as a parameter and repeatedly asks the user to guess the letters in the word. The user should be asked for a new letter while he/she has not guessed all the letters in the word. Prior to each guess the user makes the function should print the letters that have been successfully guessed, as well as the number of letters remaining to be guessed. The function should return the number of guesses taken to get all the letters. The guessing should all be done in a case-insensitive way (you can just convert to upper case everywhere).

8. Given two strings, write a function to decide whether one is a permutation of the other. For example, if you are given "cinema" and "iceman", you should return True, since these strings are anagrams.

9. Write a function that takes two strings as parameters and tests whether one string is a rotation of another, e.g., "waterbottle" and "erbottlewat". The function should return True if one string is a rotation of another and False otherwise.

Text file problems

1. Write a function called maxStock that takes a file name as a parameter (a string). Your function should find and return the stock symbol that has the highest value. The file will be structured as follows:

AAPL 402.19
IBM 26.96
ORCL 31.11
IBM 186.12

Each line has two items: a stock symbol and a price, each separated by one or more spaces. For the example file above, your function should return AAPL, since it has the highest stock price.

2. Write a function that reads a text file and returns a list of all words in the file that contain at least 3 e's. The function should take one parameter, the name of the file to read. For this function, you can't use any built-in string methods. You can only use the len function on string objects, as well as basic string indexing.

3. Say that you have a text file in which the file only contains words (you can assume that the only characters in the file are letters and spaces). Write a function called wordCount that takes two parameters: the file name (as a string) and a word to search for in the file. The function should return the number of occurrences of the word in the file. The letters in the file can be in both upper and lower case; your count should be done in a case-insensitive way (include any occurrence of the word regardless of case).

For example, if our file is named 'test.txt' and has the contents (3 lines) (not including the triple quotes at top and bottom):

```
cheese pickleS mayo mustard
pIcKlEs eggS tofu salt PEPPer
soy sauce mirin peanut BuTTer AND PiCKLES
```

wordCount('test.txt', 'pickles') should return 3.

4. Write a function that computes and returns the average length of a line in a text file. The file name should be given as a parameter to the function. The function should just return the average line length as a floating point number.

5. (This one was a past exam question.) Assume you have an input file 'data.txt' that contains, on each line, a stock trading symbol, the daily low price per share, and the daily high price per share, each separated by spaces. You can assume that each symbol will be unique. For example, the file might have the following contents:

```
AAPL 255 266
MSFT 22 24.1
GOOG 450 470.5
T 27.01 27.78
```

Write a program that finds the largest difference between the low and high share prices. The program should print the stock symbol as well as the difference between the low and high prices. For example, in the example data above, your program should identify print GOOG with a difference of 20.5.

6. Write a function that takes a file name (string) as a parameter. The file can be assumed to have the following structure:

> The file consists of English text with words and numbers. Each word or number is separated by whitespace (spaces, tabs, newlines), and may end in a punctuation character. There are numbers intermixed with the words, which may also end in punctuation characters.

For example, here is an example of the kind of text that might appear in the file:

> I once rode on a train with 55 elephants who ate 350 pounds of peanuts every 2 days. 47 clowns fed the elephants, but there were 2 slacker clowns that didn't do much. The number of lion tamers on the train was 3, but there 0 lions on the train, which was odd.

From the above text, you should find the numbers 55, 350, 2, 47, 2, and 3. You can assume

that all numbers in the file are integers.

Your task is to recover all the integers from the text and return the average and median of the numbers as a tuple. Recall that the median is the number such that 50% of the values are larger than it, and 50% of the values are smaller than it. If there are an odd number of values, there is a single median value. If there are an even number of values, the median is the average of the two middle values.

List problems

1. Write a function countevens that takes a list of integers as a parameter and returns a count of the number of even values in the list.

For example:

- countevens([2,1,2,3,4]) should return 3
- countevens([2,2,0]) should return 2

2. Write a function named sum67 that takes a list of integers as a parameter, and returns the sum of the numbers, except that it should ignore sections of numbers that start with a 6, extending to the next 7. (You can assume that every 6 will be followed by at least one 7).

For example:

- sum67([1,2,2]) should return 5
- sum67([1,2,2,6,99,99,7,1]) should return 6
- sum67([6,6,1,2,3,4,5,6,7,7]) should return 7

3. Write a function sumSquares that takes a list of numbers and computes and returns the sum of each number squared. For example, if the function is given the list [2,3,4] as a parameter, it should return 29 (4 + 9 + 16).

4. Write a function that takes a list of integers as a parameter, and returns a new list that contains only the even (non-odd) numbers from the list passed as a parameter. You should not modify the list passed in as a parameter.

5. Write a function that takes a list of integers as a parameter and removes all the odd values in the list *in place*. The function should not return anything.

6. Write a function that finds the most commonly occurring items in a list, and returns a list of those items. For example given a list [1,2,3,4,4,4,2,1,2,0,5], the function should return [4,2] since those items occur 3 times, which the maximum number of times any item occurs in the list. You can use any list methods.

7. Write a function that takes a list of positive integers and returns the sum of the odd values that are less than 7. For example, if the list [1,2,3,4,9,1,2,3,4] is passed as the parameter, the function should return 8 (1+3+1+3).

8. Write a function that takes a list of positive integers and returns a new list of integers in which only even numbers less than or equal to 7 remain. The list passed as a parameter shouldn't be modified inside the function. For example, if the list [1,2,3,4,9,1,2,3,4,9] is passed as the parameter, the function should the list [2,4,2,4].

9. Write a function that takes a list of integers as a parameter, and finds the second largest unique element in the list. To help, there are two useful functions built in to Python, max and min, that return the largest and smallest elements of a list, respectively.

You should not modify any element in the list. If you need to, however, you can make a copy of the list and modify the copy.

For example: secondLargest([1,2,3,3,4,4,5,5]) should return 4.

10. Write a function named fractionsmaller that takes a list of numbers and a single number as parameters. The function should return the fraction of values in the list that are smaller than the value given as the second parameter. You should not modify the list given as the first parameter.

For example,

```
fractionsmaller([1,4,3,2,5],3)
```

should return 0.4 since there are two values smaller than the value 3 in the list, thus 2/5ths of the elements are smaller than 3 (which is 0.4).

As another example,

```
fractionsmaller([5,2,3,6,10,2,55,3],12)
```

should return 0.875 since 7 values in the list are smaller than 12, and there are 8 values in the list, and 7/8ths is 0.875.

As one more example,

```
fractionsmaller([5,7,10],4)
```

should return 0 since there are no values smaller than 4 in the list.

11. For each of the following code segments, show what the output will be.

a)

```
def mystery1(xlist):
    while len(xlist) > 3:
```

```
            xlist.remove(max(xlist))

    mylist = [ 42, 5, 10, 6, 7, 10, 4 ]
    mystery1(mylist)
    print (mylist)
```

b)

```
    def mystery3(i, list1, list2):
        i = i + 4
        list1[-1] = 42
        list2 = list1

    i = 2
    one = [4, 10]
    two = [-1, -1]
    mystery3(i, one, two)
    print (i)
    print (one)
    print (two)
```

dictionary problems

1. Write a function `removeifmorethank` that takes two parameters: a list of integers and an integer k. The function should remove any values in the list that occur more than k times (removing **all** occurrences of that value). For example, if we call the function like:

```
    removemorethank([1,2,3,4,5,4,3,2,2,1,2], 2)
```

we should remove all occurrences of the value 2.

Restrictions: no list methods are allowed. To remove an item without using the remove method, you can say "del lst[x]" to remove an item at index x. You should modify the list in place, and the function should not return anything. All list items should remain in the same relative order when you're done. Note: if you use any dictionaries anywhere in your solution, you *can* use any dictionary methods you'd like to.

2. Texting on mobile devices has spawned a set of abbreviations due to the need for writing and responding in compact and quick ways. Create a dictionary of texting abbreviations and use it to write a teen-to-adult translator function. The function should take one parameter, which is a string to translate (like "y r u l8?"), and return a new string, which is the translated expression ("why are you late?"). Your translation dictionary obviously cannot handle *all* translations, but it should handle at least 5.

3. Write a function that takes one string as a parameter. Find and return a list of all the unique

characters in the string. Do this with (a) only using lists, (b) using a dictionary to keep track of unique letters (you still should return a list of unique letters at the end).

4. Write a function that reads a text file containing a series of dictionary words, one per line, and finds all the pairs of words in which one word is equivalent to the other word spelled backwards. The word pairs should be returned as a list of tuples (each tuple should contain a given word pair, all tuples of word pairs should be added to the list that is returned from the function).

5. You have just created a competitor to Facebook. Your app stores basic friendships in a Python dictionary such that the keys in the dictionary are user names, and the value for each key is a list of user names that are friends. Here's an example of how the friendship information is stored (your site just started so there are only four users so far!):

```
friendships = {
    'alice': ['bob'],
    'bob': ['alice','marvin','fred'],
    'fred':['bob','marvin'],
    'marvin':['bob','fred']
}
```

You can assume that all friendships are bidirectional (i.e., if a is a friend of b, then b is also a friend of a).

a. Write a function that takes the friendship dictionary as a parameter, and returns the user that has the most friends. For example, given the above dictionary, the function should return 'bob'. (For simplicity, you can assume that this function should just return 1 user with the most friends, even if there are ties for the most friends.)

b. Write a function that takes the friendship dictionary as a parameter and a user name, and returns a list of all the users that are "two friends away" from the user. (That is, all the friends of your friends.) The list should not contain any duplicates, and also not contain the name of the user passed as a parameter to the function. For example, given the above dictionary and the user 'alice', the function should return friends of alice's friends, which are just: ['marvin','fred'].