Week 8 Practice Problems

Let's start with some questions on lists.

#1. Ask the user for a size (size) and a maximum value (max_value). Write a program that randomly assigns values between 0 and max_value to a size-by-size matrix. Print the matrix to the screen, and then calculate the sum of values around the "outside" (border) of the matrix. For example, if the random matrix is:

```
8 7 2 0 4
8 4 0 2 5
1 9 4 6 5
0 2 6 9 3
1 7 3 9 5
```

then the sum around the outside is 68. You will need to import the random module and investigate the function random.randint.

#2. Write a program that asks the user for a matrix size (call it, n) and then randomly assigns the values 1 to n^2 to a n x n matrix, not repeating any values!

Then print the matrix to the screen. For example:

```
 3 25 12  4 19
 5 17 21 10  6
 1  8 16  2  7
13 22 24 11 20
23 14 15  9 18
```

Notice that you'll need some way to make sure there are no repeated numbers.

If you do this in a simple, but inefficient way, you'll notice it takes longer and longer to place the next value, or find the next location. That's OK for a 5 x 5 matrix, but won't be appropriate for a much larger (say 1000 x 1000) matrix. So then what? Try writing a version that's efficient, that takes the same amount of time to place the first number as the last. Maybe you can think of more than two different ways?

Bonus: Investigate the Python `time` module and time your different implementations.

#3. Assume a list of N elements contains a sorted list of integers. Write a function that corresponds to the following definition:

```
def binary_search (array, value)
```

The function does a binary search, and returns the index in the list where `value` is stored. If `value` isn't in the array, the function returns -1.

A binary search is a way to search a sorted list of numbers. Begin at the middle of the list and decide if the value you're looking for is in the first half of the list or the second half. If it is the first half, then look at the middle element of the first half (i.e., a quarter of thy way through the initial list) and decide if the element is in the first quarter or second quarter. If it is in the second half, look at the middle element of the second half (i.e. three-quarters of the way through the original list) to decide if you should like in the third quarter or fourth quarter. Then look halfway again, and halfway again, and …

Test your function on an array of size 100 filled in with sorted integers.

Hint: Draw a picture of what the binary search is supposed to do and while you are writing the code, use a short list (e.g., less than 10 elements) to debug it.

Now let's move to some work on sets and dictionaries.

#4. Question 1 on p. 219 of your textbook.

#5. Generalize the function from Q4 to `find_multiples`. The function should take an additional parameter, *k*, and return a set of those elements that appear *k* or more times. Note: depending on how you answered Q4, this function may require a quite different approach.

You should implement the function it in two ways: using one of the lists methods (see p. 141 of your textbook) and using a dictionary. Your code should work with a list of any type.

#6. Question 2 on p. 219 of your textbook.

#7.

a) Question 11a on p. 220 of your textbook.
b) Question 11b on p. 220 of your textbook.
c) It would be useful to automatically "sparsify" a vector. Write a function called `sparsify` that takes in a list of integers and returns a dictionary representing a sparse vector.
d) What is the output sparse vector of the following command (that uses a and c)?

    sparse_add(sparsify([0,0,1,-2,0,0]), sparsify([0,1,0,2,0,0])

If you look closely, you will notice that it is not a well-formed sparse vector. Write a function that takes in a sparse vector that may not be well-formed in this way and return a well-formed sparse vector.

#8. Write a program that asks a user to enter a string and output letter that appears the most times and the number of times it appears. You need to deal with both capital and lower-case letters.

For example:

```
Enter a word:  Bubble
The letter b appears most often: 3 times.
```

Hint: Use a dictionary.