Week 7 Practice Problems

#1. Write a function that takes an integer, n, as a parameter and returns a sorted list of n random integers from 0 to 999. As usual there are a few ways to do this. Explore the functions available to you in the `random` module and the use of the `sort()` function for lists.

#2. Write a function that takes a sorted list of integers (see Q1) and returns the median. If the list has odd length, the median is the middle number. If the list is even length, the median is the mean of the two middle numbers. Your function should work with odd and even lists. Hint: If the list is already sorted, you do not need a loop!

#3. Write a function that takes a sorted list of integers (see Q1) and returns a list with 5 elements: the minimum number, the maximum number, the median number (use the function from Q2), the arithmetic average, and the geometric average. The arithmetic average is the usual mean value that you are used to. The geometric average of n numbers is found by multiplying them all together and then taking the n^{th} root.

Note that for the minimum and maximum you should **not** use min() or max() function or a loop (think about why not). For the average you could use the sum() function. But for the geometric average you are going to need to use a loop and some mathy stuff.

#4. This is a variation of Q3 from Week 6. Reuse as much code as you can. Note that the difference is that you need to not only reverse the characters within each segment but also reverse the order in which you print out the segments. To do so, you probably need to use a list.

You are given two pieces of data:

1. A string s of length n.
2. An integer k, where k is a factor of n.

Because n is divisible by k, we can split string s into n/k sub-strings where each segment consists of a contiguous block of k characters.

Write a program that gets the user to enter a string and an integer k and prints the segments in the reversed order **and also the characters in each segment reversed**. Separate the segments by space. Example:

```
Please enter the string: University
Please enter k: 2
yt is re vi nU
```

Please enter the string: Hello!
Please enter k: 3
!ol leH

a) Break the problem into three steps: 1. Get the input. 2. Break the string into n/k substrings. 3. Reverse each sub-string. You should write at least one function to do this.
b) Looking at the examples above may give you an idea about how to solve this problem in a much easier way. What if you first reverse the whole string (you solved this problem just above for substrings) and then cleverly output it including some spaces?

#5. Create a small database of marks using nested lists and the following information. You can hard-code it in your program.

Mohamed	A, A+, C, FZ, B-
Cindy	B, B, C, A, B
Mustafa	A, A+, A+, C, C
Stefan	FZ, B, B, C, C

Write a function that takes the database and a grade as arguments and returns a list of the names of the students who got that grade in any course.

#6. The box score of a hockey game is the number of goals each team scores in each period followed by the total number of goals. We are going to represent the box score from one game in a list as follows:

```
[["MTL", [1,0,0,1]],
 ["TOR", [1,0,1,2]],
 "TOR"]
```

The box score contains 3 entries: two lists and a string.

- The first and second entries (both lists) have the same form: a string and a list of 4 integers. The string is the team name and the list is the goals scored by that team in the first, second, and third periods and then the goal total.
- The last entry is a string with the name of the team that won.

For simplicity, we will assume that there is no overtime and no ties.

You are given a list of box scores (that is a list where each element is a box score in the form defined above).

Write a function that takes a single box score and returns True if the box score is well-formed under the following rules:

- The goals for each team sum up to the total number of goals scored by the team in the box score.
- The team that is listed as winning, scored the most goals.

If either of the above is not true, return False.

Test your function by creating a list of at least 3 box scores, with at least one of each type of error and calling the function for each box score. Be careful to exactly follow the rules for the form of the box scores. Here is a list of box scores in Python format.

```
box_scores = [[["MTL", [1, 0, 0, 1]], ["TOR", [1,0,1,2]], "TOR"],
              [["VAN", [1, 2, 0, 3]], ["CGY", [1,1,0,4]], "CGY"],
              [["EDM", [18, 0, 0, 18]], ["OTT", [0,0,0,0]], "EDM"]]
```

Bonus Questions:

a. Extend the format of the box scores and your code to deal with overtime and shoot-outs. Under current NHL rules, there are no ties. You might consider adding to your checking function to make sure that there are no overtime goals if one team is ahead after regulation time and that there are no shoot-out goals if one team is ahead after regulation time or overtime. And that there can only be one goal for either team in the overtime or shoot-out.
b. Write a new function that takes a list of box scores and a team name and calculates the number of points that the team has earned. A team gets 2 points for a win and 1 point for an overtime or shoot-out loss. (No points for a loss in regulation time).
c. Adapt the problem to check the equivalent of box scores for your favorite sport: football, baseball, tennis, cricket, … (but maybe cricket will be too complicated?).