Quiz 1

This quiz is **open-book and closed-internet**. Feel free to use any physical materials you have brought, but you may not access resources online (except funprog). Proctors will be available to answer administrative questions and clarify specifications of coding problems, but they should not be relied on for coding help.

Each problem is worth 25 points, for a total of 100 points.

You *must* submit your quiz via funprog before the deadline in order to receive credit. This quiz assumes you have Python 2.7 installed on your machine.

The `resources` directory contains **Python documentation** for commonly-used data structures.

Problem 1: `second_largest`

Given a list of numbers, return the **second largest** element in the list. You may assume the list has at least two elements.

Examples:

`second_largest([1, 2, 3])` should return `2`.
`second_largest([-2, -1, 0, 1])` should return `0`.

Problem 2: `run_length_encode`

Run length encoding is a simple technique to compress sparse strings of data featuring *runs* (sequences of the same character). In this problem, each run of a character is shortened to a single character followed by the length of the run. For example, `WWWWWWWWW` is encoded by `W9`. `AB`, likewise, becomes `A1B1`.

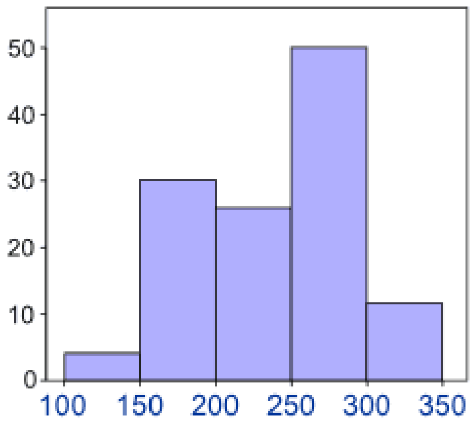
Given a string `S`, return a string that is the run length encoding of `S`. Assume only **alphabetic, capital** characters occur in the string (A-Z only).

Examples:

`run_length_encode("ABC")` should return `"A1B1C1"`
`run_length_encode("AACCA")` should return `"A2C2A1"`
`run_length_encode("ROOMMATE")` should return `"R1O2M2A1T1E1"`

Problem 3: `histogram`

In this problem, you will build a *histogram*, a common tool for analyzing numerical data. Partition the range `[low, high)` into `n` equal-sized *bins*. For example, dividing the range `[0, 10)` into two buckets yields `[0, 5)`, and `[5, 10)`. In the image below, the range `[100,350)` is divided into `5` buckets.



Given a dataset `A` (a list of integers), count how many numbers fall into each bucket. Include only numbers in the `[low, high)`; any integers in `A` that do not fall into any bucket should *not* be counted.

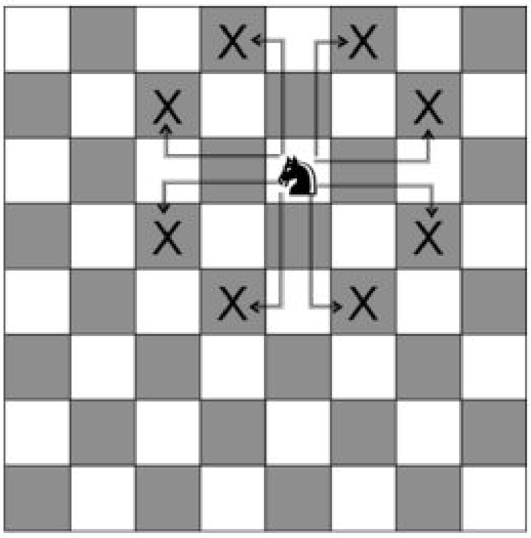
Implement `histogram( A, low, high, n )`. Return a list of counts for each bin (a list of `n` elements, where the ith element is the count for the ith bucket from the left). You may assume that `(high-low)` is a multiple of `n`.

Examples:

`histogram([1,2,3,4,5,6,7,8,9], 1, 10, 3)` should return `[3,3,3]`
`histogram([1,2,3,4,5,6,7,8,9], 0, 10, 2)` should return `[4,5]`
`histogram([1,1,1,2,3], 1, 4, 3)` should return `[3,1,1]`

Problem 4: `knight_in_two_moves`

In chess, the knight moves as follows:



We represent locations on a chessboard as coordinates of the form `[X,Y]`, where `X` and `Y` are integers between `0` and `7`. Given two locations `A` and `B` on the chessboard, write a program `knight_in_two_moves( A, B )` to determine if a knight can travel from one to the other in *exactly two moves*. Return a Boolean, `True` if the knight can indeed reach `B` from `A` in two moves, and `False` otherwise.

Examples:

`knight_in_two_moves([0,0], [4,0])` should return `True`
`knight_in_two_moves([0,0], [3,0])` should return `False`