1. **Design Tic-Tac-Tow (Medium)**

Design a Tic-tac-tow game that is played between two players on a n\*n grid. You may assume the following rules:

* 1. A move is guaranteed to be valid and is placed on an empty block.
  2. Once a winning condition is placed, no more moves are allowed.
  3. A player who succeeds in placing n of their marks in a horizontal, vertical or diagonal row wins the game.

Follow up:

Could you do better than O(n^2) per move() operation?

1. Could you trade extra space such that move() operation can be done in O(1)?
2. You need two arrays: int rows[n], int cols[n], plus two variables: diagonal, anti\_diagonal.
3. **Moving average from data stream (Easy)**

Given a stream of integers and a window size, calculate the moving average of all integers in the sliding window.

For example,

MovingAverage m = new MovingAverage(3);

m.next(1) = 1

m.next(10) = (1 + 10) / 2

m.next(3) = (1 + 10 + 3) / 3

m.next(5) = (10 + 3 + 5) / 3

1. **Longest substring with at most K distinct characters (Hard)**

Given a string, find the length of the longest substring T that contains at most K distinct characters.

For example, given s = “eceba” and k = 2, T is “ece” which it’s length is 3.

1. **Nested list weight sum (Easy)**

Given a nested list of integers, return the sum of all integers in the list weighted by their depth.

Each element in either an integer, or a list – whose elements may also be integers or other lists.

Example 1:

Given the list [[1, 1], 2, [1, 1]], return 10. (Four 1’s at depth 2, one 2 at depth 1)

Example 2:

Given the list [1, [4, [6]]], return 27. (One 1 at depth 1, one 4 at depth 2, and one 6 at depth 3; 1 + 4\*2 + 6\*3 = 27)