Innovusion Interview Coding Test

About the Test:

- Please finish the test as soon as possible after you receive it.
- You can use any computer language (python, c, c++, node …)
- Please write the code in the way that you would write as **production code.**
- The code in your answer should be executable with some test cases to demonstrate its correctness. Good and thorough test cases will earn bonus points.
- **Please submit source code with comments as well.**
- This is a open test, so you can use internet to do some research. But you should finish the problem **without other people's help.**

Problem 1: Rabbit goes home

A rabbit is in the top-left-most cell of a M x N grid. The grid width is M-cell and the height is N-cell. Some cells have snakes and other cells don't. The rabbit wants to go home, which is located at the bottom-right-most cell of the grid. It can only move rightwards or downwards (no diagonal movement) and do so one cell at a time. The rabbit cannot move to a cell that has snakes.

Question A

Write a function F that:

- inputs:
  - positive integer M
  - positive integer N
  - data structure that represents which cells have snakes
- output: how many different paths Rabbit can go home (note: please **only return the number of paths**, not the list of exact paths)

What is the time complexity and space complexity of your algorithm? Can you achieve O(M*N) for both?

Question B

In Question A, if K number of cells have snakes, can you optimize your algorithm to achieve space complexity O(max(K), min(M, N)) while keep the time complexity as O(M * N)? Please modify your code to achieve it.

Question C

list the corner test cases you want to test for question A and question B.

Question D (optional, bonus points)

If we want the function to **return all paths, not just the number of paths**, what would your algorithm be? Please implement it.

Question E (optional, bonus point)

Same as Question D, except that rabbit can move up/down/left/right, but cannot revisit a cell it has already visited, what would your algorithm be? Please implement it.

Problem 2: Find stone pair(s)

There are N number of stones, labeled as [0, N-1]. We know the weight of each of those stones. We want to find a stone pair whose weight difference is D.

Question A

Rewrite the problem as a more formal one. Something like: write a function F, the inputs of the function are … ; the output of the function is ….

Question B

- Write the function you described in your answer to question 1.
- What is the space complexity of your algorithm?
- What is the time complexity? Can you achieve O(N) for both time complexity and space complexity?
- Write a test program to test your function.

Question C (optional, bonus point)

Same as B, but this time we want to find ALL stone pairs whose weight difference is D. Please note that pair (1, 4) and pair (4, 1) are considered as the same pair, so only need to return one.