Code Challenge: Rules

This challenge is meant to measure your fundamentals. Please complete the solutions in Java. The code should compile and execute without modification, or dependency download, or machine specific settings. It is also required that you write out the tests. The minimal test set should include a test function, accepting the incoming data, and printing out the incoming data and the result for a developer to see in the console. Please do not input data from console or file, all the input should be provided in the test code.

Feel free to use any resources available to you. However, you should not rely on external libraries, nor include their source code in your submission. If you need a particular helper function, it's expected that you write it yourself and include it in your submission.

Please send your solutions by email - do not upload them to github or any other public code repository.

Code Challenge #1:

You are working on implementing a module for statistical analysis. You are given a list of numbers, where every *i-th* number represents a number of unique visitors for your website for *i-th* minute. Due to some hardware issues it might happen that a number of visitors for a certain minute cannot be determined, and thus a number representing that minute will be null. A non-empty continuous period of time for which a number of visitors for each minute was determined is called a *Monitoring Window*. A *Monitoring Window* which is not contained in any other *Monitoring Window* is called an *Interesting Monitoring Window (IMW)*. You need to design and implement a utility class(es) providing the following functionality.

The **maxSum** function should return a sub-list of a given list corresponding to an IMW with a maximum number of visitors determined. If there is more than one such IMW that exists, this function should return the longest one.

The **minAvg** function should return a sub-list of a given list corresponding to an IMW with a lowest average number of visitors. If there is more than one such IMW that exists, this function should return the earliest one.

If it’s not possible to find a required sub-list, both functions should return an empty list.

Example:

- maxSum(1, 5, null, 1, 2, 2, null, 3) → [1, 5]
- minAvg(1, 5, null, 1, 2, 2, null, 3) → [1, 2, 2]

  There are 10 Monitoring Windows: [1], [5], [1, 5], [1], [2], [2], [1, 2], [2, 2], [1, 2, 2], [3], but there are only 3 IMVs: [1, 5], [1, 2, 2], [3].

Code Challenge #2

Given an array of random integers, and a given integer **S**, determine if it is possible to select two numbers from this list, so their sum equals to **S**.

Constraints:

- **0** <= size of a given array <= **100000**
- Each value in a given array is between -**2^31 and 2^31 – 1**