Problem A:

- Repeating Cipher

https://codeforces.com/problemset/problem/1095/A

- Use a regular for loop iterating from 0 to n. Initialize a temp variable to 0, lets call it “*skip*”. We want to output the first letter, and then the next letter two after, and then the next letter three after... and so on. Add *skip* to i and increase its value by one each pass through the loop.

Problem B:

- Grade Allocation

https://codeforces.com/problemset/problem/1316/A

- Notice that the average grade of the class will not change if you add all of a student’s points to yourself. Therefore the greedy answer is to take the sum of all students’ grades and then output the minimum between that sum and the highest possible grade.

Problem C:

- Delete from the left

https://codeforces.com/problemset/problem/1005/B

- Compare each letter of both strings starting from their ends. Break the loop if they do not match. Lets have variable ***k*** representing the number of letters that they share from the end. If they do match, add 2 to ***k***. The answer is the sum of their lengths subtracted ***k***.

Problem D:

- Make them equal

https://codeforces.com/problemset/problem/1154/B

- Find the biggest and smallest values in the array. Calculate D to be (biggest - smallest) / 2. The goal value for each array element should be (smallest + D). Now iterate through the array. If array[i] + D, array[i] - D, and array[i] are all not equal to the goal, then set a bool to false. If the bool remains true, output the calculated D. If the bool is false and there are only 2 unique values in the array, output their difference, otherwise output -1.

Problem E:

- Collecting Packages

https://codeforces.com/problemset/problem/1294/B

- Store each x,y in the array as a pair. Now you have to sort the array. In order to check if there is a possible path, you must ensure that there is no pair that has a smaller x value than another pair and also has a larger y value than that other pair. This would mean that it is to the top left of the other pair which is impossible to reach with only up and right moves. If no pair meets this condition then you can find the path by starting at (0 , 0) and always going right before going up to reach the next package. This gives you the lexicographically smallest string.