Fall23 Div II Week 1 - Solution Sketches

(taken from editorials of original problems)

Problem A

(Source: Codeforces Round 886 (Div. 4) Problem A) - Topic: Intro

```java
import java.util.Scanner;

public class ProblemA {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int t = scanner.nextInt(); // Number of test cases

        while (t-- > 0) {
            int a = scanner.nextInt();
            int b = scanner.nextInt();
            int c = scanner.nextInt();

            // Check if there is a pair of digits whose sum is greater than or equal to 10
            if (a + b >= 10 || b + c >= 10 || a + c >= 10) {
                System.out.println("YES");
            } else {
                System.out.println("NO");
            }
        }
        scanner.close();
    }
}
```

Problem B

(Source: Codeforces Round 886 (Div. 4) Problem D) - Topic: Sorting

Let's calculate the maximum number of problems we can take, and the answer will be n subtracted by that count.

An arrangement that always minimizes the absolute difference between adjacent pairs is the array in sorted order. What we notice, is that if the array is sorted, we will always take a subarray (all taken elements will be consecutive).

So, the problem converts to finding the largest subarray for which $a_i - a_{i-1} \leq k$. It's easy to see that all the subarrays are totally different (don't share any intersection of elements), thus, we can maintain a count variable of the current number of elements in the current subarray, and iterate through array elements from left to right. If we currently are at i and $a_i - a_{i-1} > k$ then we just set the count to 1 since we know a new subarray starts, otherwise, we just increase our count by 1. The answer will be n subtracted by the largest value that our count has achieved.

Problem C

(Source: Codeforces Round 254 (Div. 2) Problem B) - Topic: Union Find/DSU

It's easy to find that answer is equal to 2^{n-v}, where v is the number of connected components.

Problem D

(Source: Educational Codeforces Round 11 Problem C)

Let's call the segment $[l, r]$ good if it contains no more than k zeroes. Note if segment $[l, r]$ is good than the segment $[l + 1, r]$ is also good. So we can use the method of two pointers: the first pointer is l and the second is r. Let's iterate over l from the left to the right and move r while we can (to do that we should simply maintain the number of zeroes in the current segment).