Link to Contest:

https://vjudge.net/contest/373319
Password: rccacm

Problem A:

- Sum of Round Numbers
https://codeforces.com/problemset/problem/1352/A

- For each decimal place in the number, if it is not a zero, add it to a vector that stores your answers. You can achieve this buy modding x by 10 and then dividing it by 10 to get to the next number. I used a temp variable that gets multiplied by 10 every decimal place to make sure you're adding 500 instead of 5 or 7000 instead of 7 etc.

Problem B:

- Same Parity Summands
https://codeforces.com/problemset/problem/1352/B

- Key things to note are:
    - An even or odd number times an even number will be even
    - An even number times an odd number is even
    - An odd number times an odd number is odd

- The problem is easiest when you break it up into different scenarios. N is the given number you are trying to represent, k is how many numbers you want to write that sum up to n. If k is even, write n as k-1 ones, and then have the last number be n - k + 1. For example if you are trying to represent 8 as 4 numbers of the same parity, you would have [1, 1, 1, 5] .
- If k is odd you have two scenarios.
    - If n is even you must write it as k - 1 twos, and then have the final number be n - 2*(k-1) .
    - If n is even you must write it as k - 1 ones, and then have the last value be n - k + 1, the same approach as the first idea.

- The last thing to note is this method will not work if the last number is not a positive integer. For example in the case n = 4, and k = 6 your array would look like [1, 1, 1, 1, 1, -1] which is not valid so you would output no.

Problem C:

- K-th Not Divisible by n

https://codeforces.com/problemset/problem/1352/C

- Notice that for every range of n numbers, n - 1 of those numbers will not be divisible by n. For example if n is 5, then [1, 2, 3, 4] are not divisible by 5. With this logic we will skip (k-1) / (n-1) numbers in our sequence. Add this (k-1) / (n-1) to k to find the kth number that is not divisible by n.

Problem D:

- Alice, Bob, and Candies

https://codeforces.com/problemset/problem/1352/D

- For this problem you basically simulate the whole game. Use two pointers, one called ***left*** for Alice, and one called ***right*** for Bob. On each player's turn, move their pointer towards the center until they have eaten more than the other player on their previous turn. One thing to note is that Alice makes the first turn and always only eats the first candy in the array therefore it is easier to set those values before beginning the simulation of the game.

Problem E:

- Special Elements

https://codeforces.com/problemset/problem/1352/E

- For this problem, I stored every array element and its frequency into a map. After creating this map, simply brute force over all subsegments of the array (the length can only be 8000 therefore a O(N^2) solution is okay) . Each time you update the sum of a subsegment, check if the sum exists in the map. If it does, add its frequency to the answer and remove it from the map so you do not recount it while calculating other subsegments.