Solution to CS Cup Round 3

Regular Expression (Wildcard Matching)
This is the dynamic programming pattern problem, try to approach and come up solution with recursive and enhance it by using cache or bottom up table later.

Online solution https://workat.tech/problem-solving/approach/wm/wildcard-matching

Sub Set
This is similar concept to Backtracking or DFS.
Try to sort the input first before going to implement the solution.
Think of base case as empty elements then copy it
Online solution:
https://github.com/varunu28/InterviewBit-Java-Solutions/blob/master/Backtracking/Problems/Subset.java

The Square

Firstly, we know that the figure is Square, in which all sides are equal. It is given 2 points. So, it is necessary to find distance between them, then we can find the last 2 points.

Solution Code: https://ideone.com/8389V8

The Triangle Sides

Run three nested loops from '1' to 'N' and check whether i+j, j+l and l+i are all divisible by 'K'. Increment the count if the condition is true.

Solution Code: https://ideone.com/mw22Kr

The Number 3

The idea is based on the fact that a number is multiple of 3 if and only if sum of its digits is multiple of 3 (See this for details).
One important observation used here is that the answer is at most 2 if an answer exists. So here are the only options for the function:

1. Sum of digits is already equal to 0 modulo 3. Thus, we don't have to erase any digits.
2. There exists such a digit that equals sum modulo 3. Then we just have to erase a single digit
3. All the digits are neither divisible by 3 nor equal to sum modulo 3. So two of such digits will sum up to number, which equals sum modulo 3, (2+2) mod 3=1, (1+1) mod 3=2

Solution Code: https://ideone.com/yAuy2C

The Friends

First, you have to realize that, if there is a path from one person to another along the friendship relationships, then those two people are friends. Assume that there are five people, A,B,C,D and E, and there are four pairs of friendship relationships, (A,B),(B,C),(C,D) and (D,E). Here, from the viewpoint of A, if B is a friend, then C is a friend too, and if C is a friend, D is also a friend, …and so on, and this chain makes all the people friends. Of course, other people than

A can also be friends of all other people.

Summarizing, all the people who are connected with friendship relationships are friends with each other, and those who are not connected are not friends with each other. Rather than managing them with graphs, it is better to manage them with sets.

For example, in Sample 1, there are three friendship relationships,
(1,2), (3,4) and (5,1). This is equivalent to friendships represented by two sets, (1,2,5) and

(3,4), in which any two people from the same set are friends with each other. (We will call them as "friends-sets")

Now let us return to the problem. If a pair of people from the same friends-set are placed to the same group, they will be friends with each other, so the conditions cannot be satisfied. Therefore, they should be divided into at least
K groups, where

K denotes the maximum number of members in each friends-set.

Now, if we prove that we can always divide into
K groups, then the answer will be

K. This distribution can be achieved by a simple algorithm. In fact For each friends-set, just distribute its members one by one from group 1. With this simple algorithm, there will be at most one member from the same friends-set in every group, so we can achieve the situation where friends are in the same group. Therefore, we now see that we can solve this problem by finding the friends-sets and count the maximum number of members.

To manage the friends-set, it can be solved by using a disjoint-set data structure called Union-Find. Even if you do not know Union-Find, you can still solve the problem in the similar way by applying Breadth-First Search (BFS) algorithms to inspect connected components.

Solution Code: https://ideone.com/SX0BWv

The Teleporter

Going from one city to another is basically a tree structure. Now as there will be N travels or let's say N edges so it is guaranteed that some city will be revisited.
For example:
3 2 4 1 5
starting from 1 -> 3 -> 4 -> 1 -> 3 -> 4 -> 1 -> 3 ans so on ....
So once we get from which city the revisit starts, that's a patter which will run for infinite time.

Then we can mode K with patter size because k can be huge. That mod value in patter is our answer
Note we may reach k th city before achieving patter it handled in the code see.

Solution Code: https://ideone.com/VARkCf

The Balls

It is easy to think of one operation as a set. Of the N, the number of operations that are completely included is ⌊N A + B⌋ times. The halfway operation is at most one operation, but the remainder of N divided by A + B and the magnitude relationship of A are important. If the remainder of N divided by A + B is R, then only min (R, A) is included.

Solution Code: https://ideone.com/LFN75k