Analysis
ACM ICPC Dhaka Site 2014
Author: Md Mahbubul Hasan

Problem A:
Author: Shahriar Manzoor
Alternate: Md. Towhidul Islam Talukder, Md Mahbubul Hasan
Type: Adhoc, string.
Solution: It is quite easy problem. It is already said whatever need to be done to solve this problem. However, if you have noticed the limit or the warning below the statement, you will understand that the naive solution will not work. But you need to notice that, the transformation only matters with the initial character, position of the character does not matter. So what you should do is to take initial string: “abc..z” and then apply the transformation in this 26 length string and then you will understand which character will be replaced by which character.

Problem B:
Author: Muhammed Hedayetul Islam
Alternate: Shiplu Hawlader, Md Mahbubul Hasan
Type: Adhoc, bitwise operation.
Solution: Just take pen and paper and write numbers one after one. You will understand that, for AND it is quite often that the leading digits are 1 and then remaining digits are 0. Almost same for OR. Leading digits are 0 and remaining are 1. How to understand where to stop? Well, just loop from msg to lsb of L and H. You will know how to decide.

Problem C:
Author: Md Mahbubul Hasan
Alternate: Tasnim Imran Sunny, Anindya Das
Type: DP, Adhoc, DFS, Inclusion-exclusion.
Solution: Apology for the long statement. But I think it is quite tough to explain the entire situation clearly. The solution is combination of many techniques. The main observation is to, count number of paths that are multiple of g instead of gcd g. Once you find that, you can apply inclusion-exclusion to find out the number of paths for gcd g. For finding out number of paths with multiple g, figure out how many numbers are there in [L, H] that are divisible by g in each node then count the path by some adhoc tree tree traversal or DP. Expected complexity: O(n*g + g^2). However, we can improve the inclusion-exclusion part to glogg but that does not make the problem more beautiful rather more painful.

Problem D:
Author: Shiplu Hawlader
Alternate: Rujia Liu, Md Mahbubul Hasan
Type: Linear Programming, Duality, Minimum cost maximum flow
Solution: Hardest problem of the set. I don’t want to ruin your fund solving this problem much but to let you know, the author and alternate writers spent about 7-10 days just to ensure that the intended solution is correct and then another few more days to write the “simple” solution. Some hints: write the problem in LP, take dual, notice that it is solvable by flow, solve it! My second favourite problem in the set, I really got fun and learnt many things while solving this one.

Problem E:
Author: Shahriar Manzoor
Alternate: Derek Kisman
Type: 2D-Geometry
Solution: This is pretty easy geometry problem. Just write down some equations and that’s it. The problem is really nice and clean.

Problem F:
Author: Md Mahbubul Hasan

Alternate: Derek Kisman, Rujia Liu, Shiplu Hawlader
Type: Greedy, Adhoc
Solution: My favorite problem of this set. I am not biased as it is my problem but I really enjoyed solving it. Not surprisingly, all of the author/alternate writers had different solutions. Rujia helped us to verify our greedy solution by his slow O(n^2) greedy algorithm. It also helped us to ensure we have some cases to block bad solutions. Derek Kisman had some simple solution which was implemented by Shiplu, however it had a flaw. So Shiplu fixed it by just shooting in the empty sky. We verified it by running against all the strings up to length 20 characters. My solution is case analysis with a very nice trick: “introduce one cost variable”. Suppose you have: a+a at the beginning. What should you do? Swap the last two characters to: “aa+” or spend two costs to add one more character at the beginning and one operator at the end? If you think for a while you will understand at some places it will be profitable to swap and some places it is profitable to insert. So what I did is swap, but now I will have extra variable-operator pair with only 1 cost (instead of 2). This makes the whole analysis lot simpler. I don’t want to ruin your fun solving this problem if you did not yet.

Problem G:
Author: Tasnim Imran Sunny
Alternate: Tanaeem Muhammad Moosa, Kazi Rakibul Hossain
Type: DP, Inclusion-Exclusion
Solution: This is a nice extension of derangement problem. Instead of directly counting number of permutations having at least K good pairs, think in terms of pairs. You will not consider if “doing this pair other pair may be joined too”. So you will over count in this step. Later you can remove these over counts by inclusion exclusion.

Problem H:
Author: Monirul Hasan
Alternate: Shahriar Manzoor
Type: Adhoc, Loop, Sorting
Solution: There are many approach to solve this problem. And I believe to make the problem easier the limit was quite low. One of the solutions may be to fix b and then run some binary search to find optimal position for a and c. I believe there is O(n) solution too. Instead of using binary search we can replace it by sliding window technique.

Problem I:
Author: Monirul Hasan
Alternate: Shahriar Manzoor, Muhammed Hedayetul Islam
Type: 3d-Geometry
Solution: Neat and simple problem with a very simple solution. If you have some 3d-geometry simple functions like computing volume of a mesh or may be just an arbitrary pyramid, then you dont need to worry. Otherwise its still doable. Just some very basic area/volume computation.

Problem J:
Author: Anindya Das
Alternate: Shiplu Hawlader
Type: Number theory, Adhoc, Observation
Solution: I think the problem is not as scary as it looks. If you play with some random list of numbers pretty soon you will understand that the resulting sequence should be bitonic sequence. So the rest is to figure out in how many ways can you make this bitonic sequence. That’s simple right?