Editorial by: Md Mahbubul Hasan
The author did not write all the solutions. So there might be mistakes. Feel free to point them out.

Problem A Brick Walls
Author: Monirul Hasan
Alternate: Hasnain Heickal
This is mostly case analysis and requires clever thinking to make the solution easier. One way to approach can be-
Suppose the source is always in the upper side (having greater y) than the target. So you will always go down. Now, if you do some hand simulation, you will find out some pattern. For example, suppose you are starting from (100, 100). So in the same row, the shortest distances are like: … 4, 3, 2, 1, **0**, 1, 2, 3, 4 … In y=99, the distances are like: … 4, 3, **2, 1, 2**, 3, 4, …, then in next row (y = 98), the distances are like: … 7 6 5 **4 3 4 3 4** 5 6 7 … So basically there is a portion at the center, where all the odd/even x coordinate points will have the same distance (for y = 98, 4 3 4 3 4), and then the distance will gradually increase (5 6 7…). So using this pattern you can find out some formula very easily.

Problem B Bracket Sequence
Author: Md Mahbubul Hasan
Alternate: Md Ashraful Islam
First solve “given a bracket sequence with four different brackets. Is it balanced?” The solution is quite easy, it uses “stack”. The solution idea for the main problem evolves from this. First run this *matching-using-stack* process. Then for each character we know which character matches with which one, and still the bracket sequence inside these two brackets are balanced. If there is no matching bracket for some character, that’s okay, you can leave it 0. But basically the stack matching procedure ensures you that, if there is some non-zero length of a balanced bracket sequence starting from a character, then this procedure will give you non-zero length. It may not be the optimal one, but it will give you non-zero. For example, ()(). Here stack procedure will give you: 2 0 2 0, although this is not optimal solution, but we are okay with that.

Now the second phase of the solution. We want to figure out the answer for starting from each of the characters in the given string. Suppose we know the solution for starting from all of: y+1, y+2, … n (say the length of the string is n and it is 1 indexed). Then, you can easily find the answer for starting at y. How? If the stack-process-value at y is 0, then answer at y is zero, no doubt. But if the stack-process-value at y is say z = stack-process-value[y], then we can still attach “second-phase-value[y + z]” portion from the end of the “z” length portion. And this way we can compute our final answer second-phase-value from right to left, somewhat like dp style.

Problem C Making a Team
Author: Shahriar Manzoor
Alternate: Derek Kisman

This is quite easy. Although it was solved by very few teams! First let's fix how many different people are there for the roles TL-MM-LD-LT. Minimum 1 and maximum 4. Suppose this number is x. So there are n-x people left, when n is the total number of people. So you can choose this x people in ncr(n, x) ways. And you can take rest of the n-x people in 2^(n-x) ways [if you take some from n-x people, they will be OW]. So just make sure n-x >= 0. Now only one thing left, that is, if there are x different people left to assume TL-MM-LD-LT roles how many different ways are there to assign these roles to x different people? Since x is very low, like 4, we can easily run a bruteforce solution and precalculate the number. Or may be case analysis in paper. And that's it.

Problem D Christmas Tree
Author: Shafaet Ashraf
Alternate: Hasnain Heickal
Another easy problem. Suppose you are at the root 1, and now want to decide if this node will have children (K children), if yes, which childs you will select in order to maximize the tree size. So one option is to make your current node a leaf and no more children. OR you will choose K among your children. Which K? Good question, for each child, recursively ask: if this node was the root, what would be the maximum tree size? You get this number for all the children of 1. Pick the maximum K, sum them, add it to 1 (the root 1 itself) and return. So it is nothing but recursive dp.

Problem E Leap Birthdays
Author: Hasnain Heickal
Alternate: Md Mahbubul Hasan
The easiest problem in the set. So the idea is, suppose some one was born in Y1 and we want to see how many "birth dates" were there till (and including) year Y2. Here Y1-Y2 range was quite small. So you could easily loop through this range. And for each year you could see whether the given day-month combination is valid. Well.. since the input was valid it is most likely valid, except for the case of leap year. So if the input was leap year- you just check if the year is a leap year. And that's all.

Problem F Megamind
Author: Md Ashraful Islam
Alternate: Aninda Majumder, Nafis Sadique
Okay, let's handle some case first. If E <= K * P then Hal is defeated. How many times does Megamind needs to shoot? Easy, ceil(E/P). So let's reduce E to E' = E - K * P. Now the pattern is like: increase it by R, decrease it by K * P (say S), increase by R, decrease by S or something like: +R, -S, +R, -S … So, if (R-S) is zero or positive, you can never defeat Hal. Because you can never decrease E'. So suppose, R-S is negative. Let: F = S - R. Deduct F as many times as possible from E' (easy floor(E'/F)). And for the remaining portion, yet another division with floor or ceiling.

Problem G XOR Path

Author: Md Mahbubul Hasan
Alternate: Hasnain Heickal
First observation you need to have to solve this problem is following:
Suppose dist[i] is the distance from root to i, for an arbitrary root (by distance I mean xor distance). So the distance between two nodes a and b would be: dist[a]^dist[b]. Why? Because, when you xor, the common path portion from root-to-a and root-to-b is cancelled out.

So what is the modified problem now? Given n numbers: dist[1], dist[2] … dist[n] (which is distance from an arbitrary root), for each x in the inclusive range [0, 2^16-1], how many pair of (a, b) are there so that dist[a]^dist[b] = x. We have over counted the distances like same node to same node, but that's okay, we can easily subtract some constant say n, from ans[0], once we know all ans[x] of the modified problem.

Okay, so given n numbers, how can we quickly find ans[x] where ans[x] = number of pairs among n numbers, which give xor x? So at this point, your experience/hand book/programming competition practice gives you advantage. This is a standard problem and called: Walsh-Hadamard-Transform. You can read about it in wiki or csacademy fft tutorial.

Problem H Angry Birds Transformers
Author: Shahriar Manzoor
Alternate: Hedayet Islam, Md Mahbubul Hasan
Yet another easy problem. There are many approaches to solve this problem. One way to look at this problem is, for each point (a, b), find "where should I stand in the X axis in order to see this particular point?" If you think a bit, this turns out to be: [a-b, a+b]. So if you stand at X = x where a-b <= x <= a+b, you can see point (a, b).

So now the modified problem is, given bunch of ranges [p, q]. What is the maximum overlap at any point? This is a standard problem. You can solve it by segment tree, bit, line sweep etc. An easy solution is to: for each range [p, q] produce: (p, +1) and (q + 1, -1) tuples. Then sort these tuples by the first number, if the first number is equal then by second number. Now add the second numbers of the sorted tuples one by one. What is the maximum summation? That's the answer.

Problem I Divisors
Author: Shahriar Manzoor
Alternate: Derek Kisman
I am a bit confused how should I explain the solution. Should I use some heavy artillery or should I try to explain using some basic tools. At the end I am going for the prior one. Of course we need to know: d(n) = d(p1^a1 * p2^a2 …) = (a1 + 1)(a2 + 1)...
Now, how many primes p1 are there in N!? That's another well known formula: floor(N/p1) + floor(N/p1^2) + …
However, a bit less known fact is that, these are actually sum of digits of N in p1 base (well.. except the lsb). You can find it in Knuth's concrete math book. So the problem is:

For each prime p <= N, find N in base p, now sum the digits of the number from 0 to N in base p, except the lsb. Now it is quite easy. First think what if p = 10 (I mean for decimal case, how would you solve it?). Do the same for a prime p.

Problem J Substring Sorting
Author: Hasnain Heichal
Alternate: Nafis Sadique
If K was not present, then this was Suffix Array problem. But since K is here, it is still suffix array but requires a few modification on top of it. So instead of giving you the full details, let me show you how to approach such problem.

First find the suffix array and also the adjacent lcps in suffix array. Also process the queries offline. If K = very big, then M = M'th one in the suffix array. But what if K was a little bit small? Then there would have been slight discontinuation in the suffix array. I mean, the ones having adjacent lcp greater than (or equal to) current K then, they becomes "one". And then keep doing it, once K becomes 0, everything becomes united. At each point of K, perform the query "K M" and thus compute the answer offline. Now you need some segment tree/set/priority queue etc to solve the rest of the problem. At this point, you might think that "uniting" is difficult. Then may be, going in the reverse direction is easy? I mean, initial cur_K = 0, so all of the strings are united. Then increase cur_K one by one and thus uniting adjacent groups to a single group, and when queried you need to print the M'th united group. One/Both of this approach will lead you to the final solution.

By the way, you can think how to make the solution online.

Problem K Bermuda Polygon
Author: Mehdi Rahman
Alternate: Derek Kisman
Although this looks scary but you should notice that N is very small. In plane 2d, you can find convex hull in nlogn time, but now you don't need to be that efficient. Even n^2 is not required. So just think about most naive way you can make convex hull of n points. Among many other approaches, one way is to, take every two point, draw an edge, and see if all the other points are in one side of this edge. If so, then yes this edge is part of convex hull. Since this problem is in 3d, it is actually: given 3 points (center, and two points), look at the other points and see if all those points are one side of a 3d plane. There are a few painful special cases to the problem, for example, on the edge, no point inside hull etc, but those are details. Main idea is to figure out the convex hull on the globe.