Problem A: Giveaway
Setter: Shahriar Manzoor
Alternate Writer: Monirul Hasan, Md Mahbubul Hasan
Analysis: This problem can be solved in many ways. The code template and all the numbers are already given in the statement. One may just carefully type all the numbers. Or they can just generate all the squares and cubes by iterating, storing into set/map and checking for each number if they are in the set/map. They can also use built in function to get the square root / cube root of a number. One may also notice that all these numbers are just 6th power of an integer (i.e. 1^6, 2^6, 3^6 …) and thus check them by looping.

Problem B: Game of XOR
Setter: Md Mahbubul Hasan
Alternate Writer: Kaysar Abdullah
Analysis: The main thing to notice here is that, the later generation only depends on adjacent bits of previous generation. That is, if two bits in i'th generation are say 01, then the new bits between them in later generation are always same, they don't depend on any other bit. So based on this observation one may come up with: count[0 or 1][b1][b2][i] to count number of 0 or 1 in last generation, between b1 and b2 bits when they are at the i'th generation. Once you have this table, you can evaluate number of 0s/1s between "0 DDD.." and the given positions to find out number of 0s/1s between two positions of the G'th generation.

Problem C: National Bomb Defusing Squad
Setter: Shahriar Manzoor
Alternate Writer: Muhammad Hedayet Islam
Analysis: There are a few similar ways to solve this problem. The obvious one is probably, find out all pair distances and keep them sorted in a list. And then for each query do a binary search to how many distances are there that are less than the query. Many teams did not read the last sentence of the statement (print blank line after test cases) carefully and thus got unfortunate penalty.

Problem D: Rational Grading
Setter: Shahriar Manzoor
Alternate Writer: Monirul Hasan
Analysis: This is careful implementation problem. There is nothing special about this problem. You need to know how to convert to decimal from octal and hexadecimal. And in addition you need to know about i++, ++i, --i and i--. That's all.

Problem E: Balanced String
Setter: Kazi Rakibul Hossain
Alternate Writer: Muhammad Hedayet Islam, Shiplu Hawlader, Md Mahbubul Hasan
Analysis: Kind of a greedy problem, can be easily implemented if you think in recursive way. First, it is obvious that the first character will be (and the corresponding number to it will be 1 (say x). There is another matching) to it, and the corresponding number to it will be 0 (x -1).

Now, decrement number of x and x-1 by 1 and see is there any x+1? If so, put a (for x+1 and) for x and keep it doing recursively. If there is no, x+1 then no new nested (. If while decrementing any count you see the number is already 0, then you got an invalid string. Also after the process if the length of the bracket sequence is not same as the input, then the input is invalid.

Problem F: Number of Connected Components
Setter: Syed Shahriar Manjur
Alternate Writer: Hasnain Heickal
Analysis: The main observation is, if there is a common divisor between two numbers then they are in the same component. So for each number (1 to 10^6) you can make a list- which of the given n numbers it divides. Then you can use union find or bfs to find out how many connected components are there. You can also optimize by just considering only primes between [1, 10^6].

Problem G: Extreme XOR Sum
Setter: Shafaet Ashraf
Alternate Writer: Hasnain Heickal, Md Mahbubul Hasan
Analysis: Unfortunately the judge data is a bit weak, and thus optimized n^2 solution squeezed through the time limit. The intended solution was n*sqrt(n). The main idea is to use lucas theorem for nCr % p. Since in this problem p = 2, so it is just bit wise comparison / computation. So one can precompute for first half of the bit and brute force for later half of the bit. Since number of bit is log2(n), thus one half is, log2(n)/2, and taking 2 power, it is sqrt(n). So one should precompute in n*sqrt(n) time and answer query in sqrt(n) time.

Problem H: Harmonic Matrix
Setter: Kaysar Abdullah
Alternate Writer: Md Mahbubul Hasan
Analysis: The main observation is to think about solution for 1d array. It is less costly to make a sequence with: <><><>... pattern (O(n) swap needed only). One can easily extend this idea to 2d, although not very obvious. It takes about 1.5RC swaps to solve in this way.

Problem I: In the kingdom of hirok
Setter: Anindya Das
Alternate Writer: Hasnain Heickal
Analysis: Read the statement carefully. And then it is just SCC and some probability calculation with combinatorics.

Problem J: Prime Distance
Setter: Shafaet Ashraf
Alternate Writer: Shiplu Hawlader
Analysis: This is a bit tricky problem. The main idea is to notice that, odd + odd = even. So there can not be more than one odd prime gaps. Also there can not be more than one even prime gaps. So a few cases only: only one odd prime gap, only one even prime gap or odd/even prime

gaps so that odd+even = another odd prime. So there can be at most four cells containing positive number of coins. The limit is not that big.

Problem K: 8-ball rack
Setter: Muhammad Hedayet Islam
Alternate Writer: Kazi Rakibul Hossain, Shiplu Hawlader
Analysis: This is very painful coding problem. But the idea is simple, the author's solution is DP and alternate solution is dijkstra. You just need to define the state properly and that's it. Also I believe it can be solved by A* or IDA* or backtracking with sufficient pruning.