Editorial of
Bijoy-71 Programming Contest – organized by DUETCS

Problem A – SIXteen-th December

Setter: Pronoy Debnath, **Tester:** Naeem Al Imran & Imran Mansur
Topics: Strings, Math, Number Theory

In this problem, a simple observation is that the first two rules have nothing to do with the validity of the scored runs. By following the third rule, a team can only score runs which is a multiple of six. So, the value of A – B will also have to be divisible by six to be valid. Thus, if we check the divisibility of the number by 6, we will get the answer.
But since the number is too large, the straight forward way of checking the divisibility taking modulo will not work here for the languages like C/C++. So, we will have to take input of the number as a string. The rule of a number being divisible by 6 is that, it has to be divisible by both 2 and 3 and we can sum up the digits of the number and check its divisibility by 3 by taking modulo by 3 and check the last digit of the number whether it is even or odd to check the divisibility by 2. If the number is both divisible by 2 and 3, the scored answer is Valid, otherwise it is Invalid.
Fun fact: It’s very easy to solve the problem using Python as the language supports the operation of directly taking modulo of these large numbers.
Time Complexity: O(D); D = Number of digits
Solution

Problem B – The Provost's Challenge

Setter: Emran Khan, **Tester:** Pronoy Debnath
Topics: Trees, Math

To solve this problem, just calculate the diameter of the tree formed by connecting the edges. Placing Chuppu on the middle of that diameter of the tree will be optimal to minimize the maximum distance. So, the answer will be the ceiling value of the diameter divide by 2.
Time Complexity: O(N)
Solution

Problem C – Her Demand!

Setter: Imran Mansur, **Tester & Alter:** Pronoy Debnath
Topics: Strings, Math, Hashing

To solve this problem, you can use the KMP substring search algorithm to efficiently find all positions [i1, i2, i3, …] where the car string matches a segment of the highway. For each matching position i, take the corresponding B[i] value and calculate the cumulative points for that highway. Convert both the car and highway strings to lowercase to ensure case-insensitive matching. If the total points for any highway reach or exceed C, the answer is "Yes"; otherwise, the answer is "No". Be cautious of integer overflow when summing up the points.

Time Complexity: O(N * M)
Solution

Problem D – Flag Stand

Setter: Pronoy Debnath, **Tester:** Naeem Al Imran & Imarn Mansur
Topics: Geometry, Math, Number Theory

This is a straight-forward trigonometric problem. Just calculate the Hypotenuse of the formed right angle triangle and find out the value of $\cos^{-1}$(Base / Hypotenuse). Get the answer by converting the angle to Degrees. You have to be cautious about the fact that the height of the broken part could be less than or equal to the standing part. In that case, it won't form a triangle and the answer will be 0.
Time Complexity: O(1)
Solution

Problem E – Mr. Weirdo

Setter: Pronoy Debnath, **Tester:** Naeem Al Imran & Imarn Mansur
Topics: Implementation, Math, Number Theory

To solve this problem, just calculate the number of times you have to print the message by taking the ceiling value of one-third of the given number. Use "\\n" to print "\n".
Time Complexity: O(1)
Solution

Problem F – Moom and Her Bit Set Game

Setter: Naeem Al Imran, **Tester:** Pronoy Debnath
Topics: Bitmasks, DP, Math

This problem can be solved using Dynamic Programming (DP) with a recursive approach.
State Representation:
Define the DP state as:
dp[pos][highBit]: The maximum number of set bits that can be achieved starting from position pos with the previous value having highBit set bits.
Transition:
At each position pos, you have two choices:
Skip the current element: dp[pos][highBit] = rec(pos+1, highBit)
Take the current element:
Multiply the current value a[pos] with the highBit and calculate the number of set bits (ct) in the result. Then transition to the next position with ct as the new state:
dp[pos][highBit] = rec(pos+1, ct)
The recurrence relation is:
dp[pos][highBit]=max(rec(pos+1,highBit),rec(pos+1,ct))
where **ct** is the number of set bits in a[pos] * highBit.

Base Case:
If pos >= n (beyond the array), return the current value of highBit as there are no further elements to process.
Initialization:
Start from pos = 0 with highBit = 1, as the initial number of set bits is 1.
Time Complexity: O(N * 32)
The recursive function processes each state once using memoization.
Space Complexity: O(N * 32)
Solution

Problem G – Play With Bits
Setter: Touhid Ahmed, **Tester:** Pronoy Debnath
Topics: Strings, Math, DP, Number Theory

To solve this problem, generate all subsequence of the given array and calculate the maximum xor value of all subsequences. Note that there are 2^n subsequences of an array having n values. Since the value of n will be at most 17, it is very much possible in the given time limit.
You can also use the concept of Knapsack to solve the problem in a more efficient way.
Time Complexity: $O(2^N)$
Solution

Problem H – Snake Box
Setter: Touhid Ahmed, **Tester:** Pronoy Debnath
Topics: Strings, Math, Number Theory

A simple solution of this problem is that, just count the number of odd digits in the given number taking input as a string. S. Islam will always win if the number of odd digits is even, otherwise Mr. Perseus will win.
Time Complexity: O(N)
Solution

Problem I – El Dorado
Setter: Mehedi H. Akash, **Tester & Alter:** Pronoy Debnath
Topics: Graphs, Trees, Number Theory, Data Structures, DFS and Similar

It is a straight forward algorithmic problem. To solve the problem:
1. Identify prime-numbered cities using the pre-calculated Sieve of Eratosthenes.
2. Sort all roads by their repair cost in ascending order.
3. Use Kruskal's algorithm to build the Minimum Spanning Tree (MST) for the major cities.
4. Use DSU (Disjoint Set Union) or DFS to ensure all major cities are connected, adding new roads at a flat cost if needed.
Time Complexity: N * log(N)
Solution

Problem J – 7up
Setter: Pronoy Debnath, **Tester:** Naeem Al Imran & Imran Mansur
Topics: Implementation

The solution of this problem is very much easy. You just have to find the position of the first 7 in the array, and print -1 if the array doesn't contain 7. The only thing to take care about is that you should not print any extra trailing spaces and a newline after the last line.
Complexity: O(N)
Solution