## Problem A:

**Writer: _labib**

**Approach:**

**Simply store all the url(s) in a vector and iterate through it (if possible).**

**At first declare a vector and push_back the url 'http://www.lightoj.com/'. To keep track of total pages and the current page we are visiting, declare two integer type variables, 'curr' and 'last'. 'curr' holds the index of the current page and last holds the index of the last page (basically v.size() - 1).**

**Within an infinite loop start taking input of the commands as strings.**

**For the command "QUIT": Simply break the loop.**

**For the command "VISIT": Take input of the url and push_back it to the vector and output the url.**

**For the command "FORWARD" or "BACKWARD": First check if it is possible to go backward/forward. i.e. if we increment 'curr', it will exceed 'last' or not, in another case if we decrement 'curr' it will be positive or not. If it is possible to go forward/backward, output the link stored at that index of the vector, if not, output "Ignored".**

**If the idea is still unclear, check out the code.**

**Note: This can also be solved by using stacks.**

**Code Link: https://pastebin.com/7M1J1rSv**

## Problem B:

**Writer: sivan_iut**

**Approach:**

**Mentioned in the problem,The maximum value of input n can be (10^18)..**
**Our idea is at first we will take an array and store factorial values from (1 to 20)..We are taking till 20 because factorial of 21 will be more than the highest possible value of n..**
**We are solving it through Stack as It's data structure follows last in first out rule...We will compare fact[i] with the value of n & we will push the values of i in Stack which will satisfy the condition fact[i]<=n.We will keep updating the value of n by subtracting the values of fact[i] which satisfies the condition.**

**We will do it through a loop starting from i=20 to i=0..Our target is to find out whether the final value reaches 0 or not If yes then we will just print out the answer in the form provided in question .If not then it's impossible.**

**Note:We could use vector instead of Stack..**

**If you are still having trouble,just take a look at my code..**

**Code Link:** https://pastebin.com/7yRVaFq0

## Problem C:

**Writer: iztihad110**

**Approach:**

**First declare a deque which will contain integers. For example deque<int>q.**
**If the command is "pushLeft" , then insert the given integer to the front of the deque using q.push_front() modifier and print "Pushed in left: " along with that integer. Again if the command is "pushRight" then insert the integer given with this command to the back of the deque using q.push_back() modifier and print "Pushed in right: " along with that integer.**
**Now if the size of the deque becomes equal to the given size then print "The queue is full".**
**If the command is "popLeft" , print "Popped from left: " along with the integer which was in front of the deque. Then remove an element from the front of the deque using q.pop_front() modifier. Similarly, if the command is "popRight" , print "Popped from right: " along with the integer which was at the back of the deque. Then remove an element from the back of the deque using q.pop_back() modifier.**
**Now if the size of the deque becomes 0 , print "The queue is empty".**

**Code Link:**

**https://pastebin.com/NvHbNPsy**

## Problem D:

**Writer: sivan_iut**

**Approach:**

**Well to solve this we have to be familiar with truncated cone..It varies person to person how he or she will approach till solution..From my idea to find its volume we need to find the value of r3 & a..r3 is the radius of the top part of water & a comes from the prove..Then we just need to put the values in the equation..Be cautious when you are inputting the value of pie,It needs to be accurate and also we have to print final answer 8 values after decimal point so it will be better if we use printf here.**

**Go through my code and clarify the idea ..**

**Code Link: https://pastebin.com/RJB9fGdk**

## Problem E:

**Writer: _labib**

**Approach:**

**First find the divisors of each number starting from 1 to 1000 and store them. We have to pre-calculate the whole thing.**

**(Here's a fact! No number will have more than sqrt (1000) = 32 divisors.)**

**We can easily implement this using pairs.**

**Declare a vector of pairs, and store the total number of divisors in the first position and the number itself in the second position.**

**Then sort the vector according to the conditions. i.e. Only swap the adjacent elements if the divisor count of the first element is lesser or the number itself (of the first element) is greater.**

**Now, just take input and output the number from that index of the sorted vector.**

**Code Link: https://pastebin.com/bsczz5TL**

Problem F:

**Writer: lelbaba**

**Approach:**

**The sequence is -**
**1, 12, 123, 1234, 12345, ……,12345678910, ……**

## Observation:

**Every first integer from a pack of three is not divisible by 3, the next two are divisible by 3, again the next integer is not divisible by 3.**
**For example, 1 is not divisible, 12 is divisible, 123 is divisible, but 1234 is not divisible and the next two are divisible. This pattern continues infinitely.**
**Deduction of this pattern**

## Defining a function

**Purpose: Finding the total number of multiples of three from the $1^{st}$ to $N^{th}$ number of the sequence.**
**Input: N.**
**If the number N is 2 modulo 3, the output will be (N/3)*2 +1.**
**Else the output will be (N/3)*2.**
**Note: Here, N/3 refers to the integer division, which automatically takes the floor the result. For example if N = 4, the result would be 1, not 1.33.**

## Solving the problem

**For every test case with numbers A and B given, the output would be F(B) - F(A-1).**
**Here F(B) is the total number of multiples of 3 in the sequence until the $B^{th}$ element. And F(A-1) is the total number of multiples of 3 in the sequence until the A-$1^{th}$ element. So the subtraction gives us our desired result.**

**Code Link:**

**https://pastebin.com/mALdNMRL**

## Problem G:

**Writer: lelbaba**

**Approach:**

### Observation

**The first number is huge and will not fit any datatype. So it will be stored in a string instead. Positive or negative doesn't matter while checking divisibility so it is convenient to work with absolute value and use modular arithmetic.**

### Solution

**For each test-case,**
**First scan the first number in a string st and the second number in integer n.**
**Store the length of the string in an integer len.**
**A 64 bit Integer s is taken, which will store modulo n in each step. It is initially 0.**
**An integer x is taken to store the digit in each step;**
**Starting from the first character of the string st until the end, and store the value of the first character (by subtracting 48) of the string in x. Then multiply s by 10 and add x, then take modulo n of s. (s=(s*10+x)%n)**

**After completing the loop,**
**If s equals 0, output "divisible"**
**Else output "not divisible"**

**Code Link:**
**https://pastebin.com/LgZAWz0n**

## Problem H:

**Writer: faisal_101**

**Approach:**

**In this problem you have to take four 'integers' separated by '.' in each newline for every test case.**

**To match if the two numbers are equal, you have to convert the binary number to decimal and then compare.**

**We will use**

**scanf("%lld.%lld.%lld.%lld",&a1,&b1,&c1,&d1);**

**scanf("%lld.%lld.%lld.%lld",&a2,&b2,&c2,&d2);**

**(int can also be used instead of long long)**

**to take the input for every test case so that '.' Doesn't cause any error. Then convert the four binary numbers to a new variable in decimal. Then if all four of them match then the answer is "yes", otherwise "no''.**

**the binary to decimal conversion method:**

**Separate every digit by %10 every time and then storing it by multiplying it by (2^number of steps). We have to keep the sum of those multiplied numbers. In every step we have to divide the number by 10. When the number becomes less than zero, we move on to the next number and so on.**

**Code Link:** https://pastebin.com/ec1ZcgZX

## Problem I:

**Writer:** faisal_101

**Approach:**

**We have to find the area of a parallelogram but we do not have all four coordinates of it. Thus we cannot use determinant to find the area.**

**We need to find the $4^{th}$ coordinate. For this, look at the given graph, we can see that dx=ax+(cx-bx) and dy=ay+(cy-by); (here a,b,c,d are the points and x,y are the values of that point)**

**So now we have all the required coordinates. Now we can use determinant method to find the area. But you have to keep in mind that area can't be negative. So you have to use the absolute value of the answer. Use of 'double' variables are required for precision.**

**Code Link:** https://pastebin.com/V3wyW3uT

## Problem J:

**Writer:** faisal_101

**Approach:**

**The solution is quite simple. You are given the bottom left and the top right coordinates of a rectangular area. Now we can see that the lowest value on the X axis of the land cannot be lower than that of the bottom left one and cannot be larger than that of the top right. Same goes for the Y axis as well.**

**Now that we have a general idea of the boundary of the area, we can find if a point is inside or out of that land just by checking if they breach the lowest or the highest values of (x,y) of that land.**

**Now we have to take the points at which the cows are placed and check if they are inside or not. If they are then give output "Yes" otherwise "No".**

**Code Link:** https://pastebin.com/6a68N31S

## Problem K:

**Writer: _labib**

**Approach:**

**This problem has a recursive algorithm (thus the name, "Only Recursion is Real") but there is a much simpler way, using the std::next_permutation function.**

**First, declare an array of characters of size 27 to hold the alphabets. Using a loop, then populate the array with first N capital letters.**

**Then, inside a loop, print the array and then use the next_permutation function to populate the array with the next permutation. For each iteration, keep count of the permutations and print the first K permutations only.**

**Code Link: https://pastebin.com/bZZpJk3L**

## Problem L:

**Writer: lelbaba**

**Approach:**

This problem can be solved through brute force, as well as Trie. The brute force application would be shown as the next problem already discusses Trie.
Since the concern is the prefix, the numbers would be scanned as strings.

## Solution

1. For each test case, an integer **n** is scanned which stores the total number of strings. And a **flag** is taken which will indicate our result, it is initially **1.**

2. Then all the strings are scanned and stored in a vector **v**.

3. The vector is now sorted lexicographically using the sort function. As a result, the numbers sharing prefixes are immediately next to each other.

4. Check if there is any string that is the prefix of the next string, **this loop is the consistency checking loop**. For every string until the last string, if a string is shorter than the next string, take a **string matching loop.**
   First change the value of **flag** to **0** as the first condition is met.
   If any character of the current string doesn't match with the corresponding character of the next string, the **flag** will revert back to 1 and **break the string matching loop** as it's not necessary to check the remaining characters.

5. If the **flag** remains 0 after a complete check of two strings, **break the consistency checking loop** as the data is already inconsistent.

6. Output **Yes** if flag equals **1,** otherwise output **No**, following the necessary format.

**Code Link:**

**https://pastebin.com/HGrS2pTh**

## Problem M:

**Writer: iztihad110**

**Approach:**

**In this problem we need to use “trie”. Trie is basically a data structure which stores characters of strings.**

**First declare a structure named “trie” which will contain an array next[] whose every element will point to a new node. Then declare a function trie() inside the structure where every element of the array next[] will be assigned with NULL. Initially assign 0 to a boolean type variable end_point which will indicate the end of nodes later on. We also need an integer type count variable which is also initialized with 0. At the end of the structure declare a pointer variable “root” .**
**After that declare another function whose return type will be int. This function will return 0 for ‘A’ , 1 for ‘C’, 2 for ‘G’ and 3 for ‘T’. This will be done with the help of switch-case.**
**Then declare another function in order to insert a new string in the trie. Let the name of this function be "insert" . This function won’t return anything. Then assign the variable root to another pointer variable “curr” because our root variable will be required later on. Then iterate through every character of the string and check whether that character is present in the trie or not. If it is absent then create a new node. Then update the “curr” pointer variable and keep their count in the count variable. Take the maximum value of the number of prefixes and its characters and assign it to variable.**
**Then we need another recursive function to delete all the nodes of the trie otherwise it may exceed the memory limit.**
**Inside the main function we have to create a new trie and assign it to the root variable. Then call the insert function for all n strings taken as input. Print the variable where the maximum value was assigned for each test case. After every test case delete the root.**

**Code Link:**

**https://pastebin.com/Yj6fePdK**

Problem N:

**Writer:** lelbaba

**Approach:**

## Observation

**This game can be converted into a Nim game. In n columns there are pawns that can move in both directions. However after thinking about the game it can be realized that the distance between the pawns is the key behind the game. And the player in a winning position would never pull his pawn back as that would take him to a losing position. And even if the player in the losing position moves his pawn backward, the other player can simply move his corresponding pawn forward and make the distance equal again. The game is about decreasing the distance of the pawns, and so the player who is unable to take any of his pawn forward is the loser, and the player who makes the distance of all pawns zero is the winner. And both party take turns in decreasing the distance of a specific column (similar to taking a stone from a pile in Nim game). The xor-sum of some integers is the value obtained by applying Bitwise Xor operation on all the data one after the other.**

## Solution

**For each test case,**
**Take an integer s which would contain the nim-sum. It is initially 0.**
**Scan the number of rows/columns in integer n.**
**Then for each column, scan the position of each of the white pawns in an integer array arr[i].**
**Then for each column, scan the position of each of the black pawns in an integer x and update each element of arr so that they contain the initial distance between the two pawns in each column (arr[i]=x-arr[i]). Simultaneously take the xor-sum and store it in s using bitwise xor operation (s^=arr[i]). After finishing the process for each column, s will contain the xor-sum of the distances.**
**If s equals 0, black wins. So output "black wins".**
**Else white wins, output "white wins".**

**Code Link:**

https://pastebin.com/Z452bKWY

## Problem O:

**Writer: iztihad110**

**Approach:**

**This is basically a Nim game where both of the players are taking stones from the piles in each turn. The motive of each of the players is to decrease the amount of stones in each pile as much as possible. The player who takes the last stone from the piles, eventually wins the game. The elements of our given matrix indicates the number of stones in each pile. The XOR sum of the number of stones in each pile will determine the winner of the game. That is why we need bitwise XOR operator(^) to solve this problem.**

**First we need to add the elements of each row of the given matrix and put the sum of each row in a vector. Then we use bitwise XOR between the elements of the vector and a variable which is initially assigned with 0 and assign our XOR result to that variable. After XOR operations if the value of that variable becomes 0 then print "Bob" otherwise print "Alice" .**

**Code Link:**

https://pastebin.com/UzyEdBvJ