Mathura – Kanpur Contest Problems

(11 Problems)

09th April, 2023

ICPC Mathura-Kanpur Programming Contest
April 09, 2023

Instructions

There are **Eleven (11)** problems (20 Pages) for each team to be completed in **five hours**. Standard Input and Output files are to be used for each problem. If you test your program using CodeDrills platform, it will automatically redirect input from the sample input file to your program. Output must correspond exactly to the provided sample output format, including (mis)spelling and spacing. Multiple spaces will not be used in any of the judges' output, except where explicitly stated.

Your solution to any problem should be submitted for judging using the CodeDrills platform. Once you have submitted the solution, it will reach the CodeDrills judge and the result will be shown. The judgement may be "Correct Answer", meaning that your submission was judged to be correct. Otherwise you will get a message indicating the problem with your program. For example, the message may be "Wrong Answer", "Compilation Error", "Runtime Error", "Time Limit Exceeded" etc.

You can submit only one source file.

You can use any of the standard library functions that your chosen programming language provides. In addition, you can use the math library in C/C++. You cannot use any other library that requires an extra flag to be passed to the compiler command. If you do this, the judges will probably find a code "compilation error" in your program. Your program is not permitted to invoke any external programs. *Violation of this rule may lead to disqualification from the contest.*

Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required. The judges will only test whether the input output behaviour of your program is correct or not. The regional contest director and judges are empowered to adjust for or adjudicate unforeseen events and conditions. Their decisions are final.

Teams are ranked according to the most problems solved. Teams who solve the same number of problems are ranked by least total time. The total time is the sum of the time consumed for each problem solved. The time consumed for a solved problem is the time elapsed from the beginning of the contest to the submittal of the accepted run plus 20 penalty minutes for every rejected run for that problem regardless of submittal time. There is no time consumed for a problem that is not solved.

Teams are allowed to bring 25 pages of printed materials stamped by the authority in the contest area. Further, the team is not allowed to discuss/ talk with any other team by any means whatsoever, during the contest period. *Any such attempt, if it is detected, may lead to immediate disqualification of all the teams involved.*

Problem A

Binary Digit Escaping

Let F be a procedure that transforms a binary number X to Y. In F we go through digits from the Most Significant Bit (MSB) of X to its Least Significant Bit (LSB) and perform following operations:

- If MSB is 0, the transformation fails unless X is 0.
- If it ever finds two consecutive 0s in X, the transformation fails.
- If it finds a 1, it skips this 1, appends the next digit of X in the right side of Y. Then we again start this scanning procedure from the next digit. However If there is no digit following the "skipping 1", the transformation fails.
- At the end, if Y has the MSB 0 the transformation fails unless Y is 0.

Following is the pseudocode for F.

```
// Binary is a type containing a binary number. It may have
// multiple leading 0s (for example: 0001101). This type has
// following functions/properties:
//   - value(): returns the decimal value of the number in X
//   - length(): returns the length of the binary number it holds.
//     For example: length of 0011 is 4, 101 is 3.
//   - append(i): appends i to the end of the binary number this
//     variable is holding.
//   - (index operation) X[i]: returns the i'th most significant
//     digit (0-indexed). That is, the MSB is at the 0'th index, and
//     LSB is at "length() - 1"th index.
//
// fail(): is a function that prematurely terminates the F
// without returning any value from it. It means the transformation
// failed.
Binary F(Binary X) {
  if (X.length() == 0) fail();  // empty X.
  if (X.value() == 0 and X.length() == 1) return 0;
  if (X[0] == 0) fail();  // X is not 0 but leading 0 found.
  for (i = 1; i < X.length(); i += 1) {
    if (X[i - 1] == 0 && X[i] == 0) {
      fail();  // two consecutive 0s found.
    }
  }

  Binary Y;  // initially Y is empty.
  // This is a for loop where i's value goes from 0 to X.leng() - 1.
  for (int i = 0; i < X.length(); i += 1) {
    if (X[i] == 0) {
      Y.append(0);
    }
    else {
```

```
      // X[i] == '1'. This is a "skipping 1".
      // First make sure we have another digit after this.
      if (i == X.length() - 1) {
        // no digit after "skipping 1".
        fail();
      }
      i += 1;  // go to the next digit
      Y.append(X[i]);  // append this digit to Y.
    }
  }

  if (Y.value() == 0 && Y.length() == 1) {
    return Y;  // Here Y = 0.
  }
  if (Y[0] == 0) fail();  // Y is not 0 but has leading 0
  return Y;  // Y is the valid transformation of X.
}
```

Here are some examples:

X	Result of F(X)
1010	Transformation fails, since after the transformation we get 00 which has leading 0.
111	Transformation fails, since we don't have a digit after getting a "skipping 1"
111010	F(X) = 100. Note, it's okay if there are consecutive 0s in Y.
11010	F(X) = 100.
100	Transformation fails, since there are consecutive 0s in X.

You are given two positive integers L and H. Transform all X's of length at least L and at most H. How many distinct Y's can be found?

For example, L = H = 2 gives us two valid distinct numbers. There are four valid X's with L = H = 2 constraints. Transformation fails for 00 and 01. F(10) = 0 and F(11) = 1.

Please note, for appropriate values of L and H, suppose 111010 and 11010 could be candidates of X. However both of them produce Y = F(X) = 100. We are only interested in the count of unique Y's.

Input

First line contains a positive integer T, the number of test cases. Hence follows T lines with three integers each: L H P.

Constraints

- $1 \le L \le H \le 1{,}000{,}000$
- Sum of H over all test cases is at most 1,000,000.
- P is a prime
- $1{,}000{,}000{,}000 \le P \le 2{,}000{,}000{,}000$.

Output

For each case, output the case number and the answer modulo P.

Sample Input	Output for Sample Input
2 2 2 1000000007 2 10 1000000007	Case 1: 2 Case 2: 48

Problem B

Little Seherish and Tree

Little Seheris was travelling to visit her grandfather's place Jashore from Dhaka.

Meanwhile, she was bored because of the long journey. To overcome her boredom, her mother Modhu made a game for her to play together.

The game is simple: there will be a tree of N nodes and each node has an integer value.

They can perform some operations on that tree by turns.

In one operation, one will choose two adjacent nodes, u and v having values A[u] and A[v]. Then choose an integer value X > 1 that divides both these numbers.

Then, replace these values A[u] = A[u] / X and A[v] = A[v] / X.

Now, for a given tree, you have to find the maximum number of operations that can be performed.

Input

Input starts with an integer **T (1 ≤ T ≤ 10)** denoting the number of test cases.

Each case starts with an integer **N (1 ≤ N ≤ 500)** denoting the total number of nodes in the tree. The nodes are numbered from **1 to N**.

Next line will contain an array **A** of **N** integers, **A[i] (1 ≤ A[i] ≤ 10^9)** is the value of i'th node.

Each of the next **N-1** lines contains two integers **u** and **v (1 ≤ u,v ≤ N and u ≠ v)** denoting nodes **u** and **v** are connected by an edge.

Output

For each case, print the case number and the result in a single line.

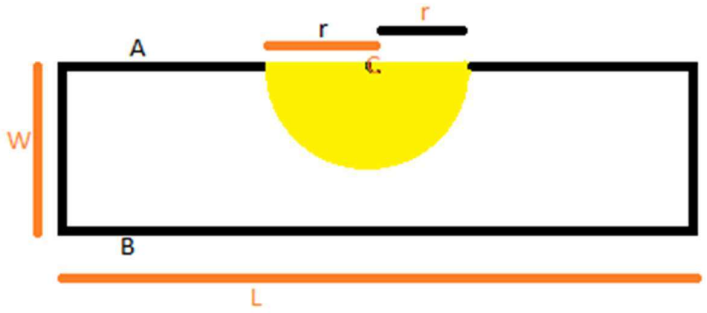
Sample Input	Output for Sample Input
2 4 2 14 6 10 1 2 3 1 2 4 3 7 2 5	Case 1: 2 Case 2: 0

1 2 1 3	

Problem C

Magic vs Geometry

Dr Strange is in a plane in a different dimension whose length is L and width is W. Now he wants to post some magical circular cast in the plane but can put the cast only on side A in such a way that the center of the cast is on line A. Cast radius r means it can take control up to r meter around its projection point (C is the projection point in the figure and the yellow portion is the coverage area). For side B he can not directly cast the magic but he can use another special magic that will fold the plane and line B will be over line A and all the cast on line A will replicate on line B.



Dr. Strange knows the values of L, W and r. As he already lost to Spider-Man because of geometry, he wants your help to determine the minimum number of magics he needs to cover up the whole board. You should assume that he has a finite number of casts.

Input

At first, there will be an integer T (1 ≤ T ≤ 100), which is the number of test cases. Each test case contains 3 integers L, W and r (1 ≤ L, W, r ≤ 1000).

Output

For each case, print the case number and the minimum number of magics Dr. Strange needs to cover up the board if possible otherwise print “Impossible”. See the samples for exact formatting.

Sample Input	Output for Sample Input
2 1 1 10 10 10 1	Case 1: 1 Case 2: Impossible

Problem D

K-th Number

In data structure, different sorting techniques are used for arranging and manipulating data in an organised way. Different sorting techniques are used in different circumstances as each of the algorithms have their own strengths and weaknesses. Firstly, let us know some commonly used sorting technique names along with their inventors. ***Mergesort*** was invented by ***John von Neumann*** while ***quicksort*** was invented by ***Tony Hoare***. ***J. W. J. Williams*** invented ***heapsort*** and ***Harold H. Seward*** invented ***radix sort***. ***Shell sort*** was named after ***D. L. Shell***. ***Vuasort*** was introduced by ***Jack Dumb***.

Wait, have you ever heard the name of Vua Sort? Ohh! It is a pretty new technique of sorting used for creating and solving different mathematical puzzles. Working procedure of ***Vua Sort*** algorithm is as follows:

```
procedure VuaSort(A be an array of integers indexed from 1 to n )
    for i := 1 to n - 1 inclusive do
        for j := i+1 to n  inclusive do
           if digitSum(A[i]) > digitSum(A[j]) swap(A[i], A[j])
           else if digitSum(A[i]) == digitSum(A[j])
              if highestDigit(A[i]) > highestDigit(A[j])swap(A[i], A[j])
              else if highestDigit(A[i]) == highestDigit(A[j])
                   if lowestDigit(A[i])>lowestDigit(A[j])swap(A[i],A[j])
                   else if lowestDigit(A[i])===lowestDigit(A[j])
                       if A[i]>A[j]swap(A[i],A[j])
                       end if
                   end if
              end if
           end if
         end for
    end for
end procedure
```

For clarity, digitSum(142) = Summation of digits of 142. digitSum(142) = 1+4+2 = 7.

highestDigit(142)= highest valued digit amongst {1, 4, 2} = 4.

lowestDigit(142) = lowest valued digit amongst {1, 4, 2} = 1.

In this problem you are given an array consisting of all the integers from L to R inclusive. You have to sort them using the Vua sort algorithm and finally answer the K^{th} element in the sorted sequence. Note that numbers with leading zeros are not valid. For example: 42 is a valid integer but 042 is not valid as leading zeros are not allowed.

Input

The first line of the input contains an integer **T**, denoting the number of test cases**.** Each of the next **T** lines will contain three integers, **L**, **R** and **K.**

Constraints

- **1 ≤ T≤ 50000**
- **1 ≤ L, K, R ≤ 10^{10}**
- **L ≤ R and K ≤ R-L+1**

Output

For each test case, print the case number and the k^{th} integer in the sorted sequence**.**

Sample Input	Output for Sample Input
2 1 10 5 20 100 45	Case 1: 4 Case 2: 91

Problem E

Diameter Division

The country of Treeland consists of **n** cities and **n-1** roads. Each road is bidirectional and connects two distinct cities. You can travel between any pair of cities using the roads. In other words, the cities of Treeland form a tree.

For administrative purposes, Treeland will be divided into **k** districts. Every node should belong to exactly one district. The cost of managing one district is the maximum distance between any two nodes in that district. For efficiency, the division should be done in a way such that maximum cost over all divisions is minimised.

Unfortunately, the value of **k** has not been decided yet. So, you have to find the minimum cost of division if we divide Treeland into **k** districts for each **k** from **1** to **n**.

Input

The first line contains **T**, the number of test cases.

Each case starts with a line containing a single integer, **n**, the number of cities. **n-1** lines follow each containing two integers **u** and **v**, describing a road between city **u** and city **v**.

Output

For each case, output a single line containing **n** integers, the **i** th integer should be the minimum cost for dividing into **i** districts.

Constraints

- **1 ≤ T ≤ 100**
- **2 ≤ n ≤ 30000**
- Sum of n over all cases does not exceed 30000.
- **1 ≤ u, v ≤ n.**
- **The input describes a valid tree.**

Sample Input	Output for Sample Input
2 5 1 2 2 3 2 4 1 5 9 1 2 2 5 5 3 4 7 7 1 3 9	3 2 1 1 0 6 3 2 2 1 1 1 1 0

3 8 6 1	

Problem F

Robot and the Doors

You are given a N×N grid. A robot is currently at the cell (1, 1) of the grid. Your task is to assist the robot in reaching the cell (N, N) using a remote control. The robot can only move up, down, left, or right, with each move incurring a cost of 1. The cells on the grid may contain:

- '.' indicating an empty cell.
- '#' which means there is a barrier in this cell.
- One capital letter (from A to Z) indicating a cell with a door.
- One small letter (from a to z) meaning the cell contains a key.

The robot can move to any adjacent cell that is empty or has a key on it. It can also move to a cell containing a door if it has already collected the relevant key to open the door. A single door can only be opened using the key labelled with the associated small letter (e.g: door "A" can be opened using key "a", door "B" can be opened using key "b" and so on). There will be only one door in the grid with a specific label, however, duplicate keys may be present.

The robot must collect keys in sequential order, first key "a", then key "b", and so on. If the robot reaches a cell with a key but has not yet collected all the previous keys in the sequence, it will leave it for later. Your task is to find the minimum cost for the robot to reach cell (N, N).

N.B: The top leftmost cell of the grid is (1, 1) and the bottom rightmost cell is (N, N). There will be no barrier in the cell (1, 1) as the robot is already at the cell

Input

In the input, the first line consists of an integer **T** (**1 ≤ T ≤ 10**) that represents the number of test cases.

The first line of each test case contains an integer **N (1 ≤ N ≤ 200)**, indicating that the grid given will be of size **N×N**. Each of the following N lines each contain a string of **N** characters, where each character describes a cell in the grid.

Output

For each test case, print the minimum cost in the format "**Case X: Y**" (without quotes) where **X** is the test case number and **Y** denotes the answer. If the robot can't reach the destination, then print "**Case X: Impossible**" (without quotes). Please see the sample output for more understanding.

Sample Input	Output for Sample Input
3 5 ....#.a ....# ...A. 3 ... ..B .A. 3 ... .#	Case 1: 10 Case 2: Impossible Case 3: Impossible

Problem G

Marriage vs Gym

Mr. X is going to marry Miss Y. Miss Y wants to have good photos in the ceremony so she asks Mr. X to go to the gym to have a good body.

The marriage is after **M** days. He wants to have a fixed gym routine. He knows that he will ate x_i calories for the next M days.

Mr. X has L calories and wants to limit his calories to K calories on the ceremony day, which is M days away. He will burn N calories daily via daily activities. He plans to burn additional calories at the gym, but must have a minimum of C calories after the gym to maintain proper bodily function.

Now he asks you to find out the number of calories he needs to burn daily so that he has at most **K** calories on the ceremony day. **He wants to see himself as fit as possible.**

N.B.: i^{th} day start calories + x_i - N >= C assuming that he never workout for any day.

Input

At first, there will be an integer **T (1 ≤ T ≤ 100)**, the number of test cases.

For each test case start with 5 integers **M (1 ≤ M ≤ 100000)**, **K**, **C**, **N** and **L (1 ≤ N < C < K, L ≤ 100000).**

Next line will contain **M** integer x_i **(1 ≤ i ≤ M** and **1 ≤** x_i **≤ 100000)** representing the calorie consumption of **i-th** day after today.

Output

For each case, print the number of calories Mr. X needs to burn every day to have at most **K** calories on the ceremony day to see himself as slim as possible if possible otherwise print "-1" (without quotes). See the samples for exact formatting.

Errors less than 10^{-6} will be ignored

Sample Input	Output for Sample Input
2 2 100 50 25 120 10 20 2 85 51 35 67 36 49	25 15.5

Problem H

Factorization Game

Alice and Bob play a factorization game. Initially, an integer **X** is given in the game. Alice starts the game and tries to factorise **X** into two integers **A** and **B** such that **A** * **B** = **X** and both **A** and **B** are greater than **1**. Then, Bob picks one number from **A** and **B** and removes the other. Let's say **Y** is the integer Bob chooses from **A** and **B**. Now, in the same way, Bob tries to factorise **Y** into two integers **C** and **D** such that **C** * **D** = **Y** and both **C** and **D** are greater than **1**. And the game continues. The player who is unable to factorise according to the rule will lose the game. If both players play the game optimally, for a given integer **X**, find out who will win the game.

Input

The first line will contain a single integer **T**. Each test case will have a single integer **X**, denoting the integer that is given in the game initially.

Constraints

- $1 \leq T \leq 5 \times 10^4$
- $1 \leq X \leq 10^9$

Output

For each case, print one line with "**Case <x>: <y>**", where **x** is the case number, **y** is "**Alice**" or "**Bob**" indicating the winner of the game.

Sample Input	Output for Sample Input
2 16 30	Case 1: Alice Case 2: Bob

Problem I

Boring Mode

You are given an array **A** of **N** integers. You have to divide the integers into **K (≤ N) non-empty** subsets such that the sum of the Mode of **K** subsets is maximised.

The Mode of an array of integers is defined as an integer that occurs the most number of times. **If there are more than one such integer, you can consider any of them as the Mode.**

For example, The Mode of [2, 3, 2, 1] is 2.

The Modes of [2, 3, 2, 3, 1] are 2 and 3. You can consider either of them as the Mode of these integers.

Input

The first line will contain a single integer **T (1 ≤ T ≤ 100)** denoting the number of test cases. The description of the test cases follows. Each test case will have two lines of input.

- The first line contains two space-separated integers **N** (**1 ≤ N ≤ 200**) and **K** (**1 ≤ K ≤ N**).
- The second line contains **N** space-separated integers $A_1, A_2, ..., A_N$ denoting the elements of the array. (**$0 \le A_i \le 10^9$ for $1 \le i \le N$**)

Output

For each test case, print on a new line, the maximum sum of the Mode of **K** subsets.

Sample Input	Output for Sample Input
2 4 3 1 2 3 1 4 2 1 2 3 1	6 5

Explanation:

For 1st case, possible optimal division: {1}, {3, 1}, {2} and the sum of modes = 1 + 3 + 2 = 6.

For the 2nd case, possible optimal division: {3, 1}, {2, 1} and the sum of modes = 3 + 2 = 5.

Problem J

Road Trip

In our world, countries follow either the left-hand or the right-hand traffic rule. That means, all vehicles on the roads must stick to a specific side while driving. In general, vehicles built for a particular side traffic system have the driving seat on the opposite side. Cars that will follow a left-hand traffic rule have the steering wheel on the right seat, and vice versa.

But in the strange land of *i12345* (situated somewhere around the universe), there are both left-hand side and right-hand side roads. All drivers there are skilled enough to drive on any of those roads. And just like large aircrafts in our world, all cars in *i12345* have identical driving gears on both front seats. People have their own choices on which side they want to sit in for driving, and also, which type of road they prefer. You see, that's the reason for them to have such a system!

There are **N** road junctions and **M** roads in the land of *i12345*. Two friends, *23l* and *145r* are going out for a drive from one of those junctions, **S** to another one, **D**. Both of them want to drive. *23l* prefers the left-hand side roads and *145r* prefers the other ones. They have decided to keep altering the type of road they will use until they finally stop at the destination so that both of them can drive. For example, they can take a route where the first road is of left-side drive, the second one is of right-side, then the next one is left-side again and so on. It doesn't matter which type they choose as the first one, that means ***lrl***… or ***rlr***… both are ok. On top of that, as the whole land is crazy with numbers, they want to pick a route such that the total distance they have travelled becomes a multiple of **K**. Your job is to generate the shortest route (route with the minimum distance) for them from their starting location, S to the destination, D which satisfies both requirements.

Input

The first line of input contains three integers **N** (≤ 1000) and **M** (≤ 10000) and **K** (1 ≤ K ≤ 3). M lines will follow each containing four integers **U**, **V** (1 ≤ U, V ≤ N), **W** (1 ≤ W ≤ 10000) and **H** (1 or 2). representing a bidirectional road between U and V with distance W. This road can be used for going from U to V or otherwise. H=1 means the road has a left-hand traffic rule, and H=2 means right.

The next line will contain an integer **Q** (≤ 1000) denoting the number of queries (numbered as 1,2,...Q). Then there will be Q lines each containing two integers **S** and **D** (1 ≤ S, D ≤ N).

Output

Print Q lines in the following format:
The query number at first. Then the distance of shortest route from S to D satisfying the constraints or the word "no such route" without quotes. Then a new line.

Sample Input	Output for Sample Input
4 5 3 1 2 3 2 1 3 3 2 2 3 1 1 2 4 5 2 3 4 2 2 2 1 4 2 3	1 6 2 no such route

Problem K

Gotta Catch 'Em All

Alice and Bob are two young trainers who are passionate about the world of Pokémon. They have been training and battling with their Pokémon for years, but they have never played a card game before. One day, they hear about a local tournament that is going to take place in their town. The tournament is a card game competition where each player will have N Pokémon cards with distinct power levels.

Excited about the tournament, Alice and Bob start practising with their Pokémon cards. They spend hours studying the different power levels of their cards. They even create new strategies to defeat their opponents.

The day of the tournament arrives, and Alice and Bob are ready to compete against each other. The tournament has N rounds, and in each round, they will each play one card, and the card with the higher power level will win the round. The winner of the tournament will be the one who wins the most rounds among the N rounds. Please note, Alice and Bob both know about their opponent's cards, and in each round, Alice will show her card first. They can play each one of their cards only once.

As a fan of Pokémon battles, you have also come to witness this epic battle. You are so excited and can't wait that long to know who will be the winner. Hence, you decide to find out how many rounds will Alice win if both players play optimally.

Input

The first line of the input contains a single integer **T ($1 \le T \le 10^5$)**, denoting the number of test cases.

For each test case, the first line contains a single integer **N ($1 \le N \le 2*10^5$)**, denoting the number of cards each player has.

The second line of each test case contains **N** integers **$a_1,a_2,\ldots,a_n$ ($1 \le a_i \le 10^9$)**, denoting the power of Alice's Pokémon cards.

The third line of each test case contains **N** integers **$b_1,b_2,\ldots,b_n$ ($1 \le b_i \le 10^9$)**, denoting the power of Bob's Pokémon cards.

Powers of all of the **2*N** cards are distinct. Besides, it is guaranteed that the sum of **N** over all test cases does not exceed **$2*10^5$**.

Output

For each test case, print a single line in the format **"Case X: Y"** (without quotes), where **X** is the test case number and **Y** is the maximum number of rounds Alice can win if both players play optimally.

Sample Input	Output for Sample Input
2 1 10 100 4 5 10 8 3 1 6 2 11	Case 1: 0 Case 2: 2