Relic Discovery

Time Limit: 2000/1000 MS (Java/Others) Memory Limit: 65536/65536 K (Java/Others)
Total Submission(s): 221 Accepted Submission(s): 171

Problem Description

Recently, paleoanthropologists have found historical remains on an island in the Atlantic Ocean. The most inspiring thing is that they excavated in a magnificent cave and found that it was a huge tomb. Inside the construction,researchers identified a large number of skeletons, and funeral objects including stone axe, livestock bones and murals. Now, all items have been sorted, and they can be divided into N types. After they were checked attentively, you are told that there are A_i items of the i-th type. Further more, each item of the i-th type requires B_i million dollars for transportation, analysis, and preservation averagely. As your job, you need to calculate the total expenditure.

Input

The first line of input contains an integer T which is the number of test cases. For each test case, the first line contains an integer N which is the number of types. In the next N lines, the i-th line contains two numbers A_i and B_i as described above. All numbers are positive integers and less than 101.

Output

For each case, output one integer, the total expenditure in million dollars.

Sample Input

```
1
2
1 2
3 4
```

Sample Output

```
14
```

Statistic | Submit | Clarifications | Back

Pocket Cube

Time Limit: 2000/1000 MS (Java/Others) Memory Limit: 65536/65536 K (Java/Others)
Total Submission(s): 291 Accepted Submission(s): 70

Problem Description

The Pocket Cube, also known as the Mini Cube or the Ice Cube, is the 2 × 2 × 2 equivalence of a Rubik's Cube.
The cube consists of 8 pieces, all corners.

Each piece is labeled by a three dimensional coordinate (h, k, l) where h, k, l ∈ {0, 1}. Each of the six faces owns four small faces filled with a positive integer.
For each step, you can choose a certain face and turn the face ninety degrees clockwise or counterclockwise.
You should judge that if one can restore the pocket cube in one step. We say a pocket cube has been restored if each face owns four same integers.

Input

The first line of input contains one integer N(N ≤ 30) which is the number of test cases.
For each test case, the first line describes the top face of the pocket cube, which is the common 2 × 2 face of pieces labelled by (0, 0, 1),(0, 1, 1),(1, 0, 1),(1, 1, 1). Four integers are given corresponding to the above pieces.
The second line describes the front face, the common face of (1, 0, 1),(1, 1, 1),(1, 0, 0),(1, 1, 0). Four integers are given corresponding to the above pieces.
The third line describes the bottom face, the common face of (1, 0, 0),(1, 1, 0),(0, 0, 0),(0, 1, 0). Four integers are given corresponding to the above pieces.
The fourth line describes the back face, the common face of (0, 0, 0),(0, 1, 0),(0, 0, 1),(0, 1, 1). Four integers are given corresponding to the above pieces.
The fifth line describes the left face, the common face of (0, 0, 0),(0, 0, 1),(1, 0, 0),(1, 0, 1). Four integers are given corresponding to the above pieces.
The six line describes the right face, the common face of (0, 1, 1),(0, 1, 0),(1, 1, 1),(1, 1, 0). Four integers are given corresponding to the above pieces.
In other words, each test case contains 24 integers a, b, c to x. You can flat the surface to get the surface development as follows.

```
+ - + - + - + - + - + - +
| q | r | a | b | u | v |
+ - + - + - + - + - + - +
| s | t | c | d | w | x |
+ - + - + - + - + - + - +
        | e | f |
        + - + - +
        | g | h |
        + - + - +
        | i | j |
        + - + - +
        | k | l |
        + - + - +
        | m | n |
        + - + - +
        | o | p |
        + - + - +
```

Output

For each test case, output YES if can be restored in one step, otherwise output NO.

Sample Input

```
4
1 1 1 1
2 2 2 2
3 3 3 3
4 4 4 4
5 5 5 5
6 6 6 6
6 6 6 6
1 1 1 1
2 2 2 2
3 3 3 3
5 5 5 5
4 4 4 4
1 4 1 4
2 1 2 1
3 2 3 2
4 3 4 3
```

```
5 5 5 5
6 6 6 6
1 3 1 3
2 4 2 4
3 1 3 1
4 2 4 2
5 5 5 5
6 6 6 6
```

Sample Output

```
YES
YES
YES
NO
```

Statistic | Submit | Clarifications | Back

Home | Top **Administration**

Pocky

Time Limit: 2000/1000 MS (Java/Others) Memory Limit: 65536/65536 K (Java/Others)
Total Submission(s): 274 Accepted Submission(s): 120

Problem Description

Let's talking about something of eating a pocky. Here is a Decorer Pocky, with colorful decorative stripes in the coating, of length L.
While the length of remaining pocky is longer than d, we perform the following procedure. We break the pocky at any point on it in an equal possibility and this will divide the remaining pocky into two parts. Take the left part and eat it. When it is not longer than d, we do not repeat this procedure.
Now we want to know the expected number of times we should repeat the procedure above. Round it to 6 decimal places behind the decimal point.

Input

The first line of input contains an integer N which is the number of test cases. Each of the N lines contains two float-numbers L and d respectively with at most 5 decimal places behind the decimal point where $1 \le d, L \le 150$.

Output

For each test case, output the expected number of times rounded to 6 decimal places behind the decimal point in a line.

Sample Input

```
6
1.0 1.0
2.0 1.0
4.0 1.0
8.0 1.0
16.0 1.0
7.00 3.00
```

Sample Output

```
0.000000
1.693147
2.386294
3.079442
3.772589
1.847298
```

Statistic | Submit | Clarifications | Back

Lucky Coins

Time Limit: 2000/1000 MS (Java/Others) Memory Limit: 65536/65536 K (Java/Others)
Total Submission(s): 83 Accepted Submission(s): 19

Problem Description

Bob has collected a lot of coins in different kinds. He wants to know which kind of coins is lucky. He finds out a lucky kind of coins by the following way. He tosses all the coins simultaneously, and then removes the coins that come up tails. He then tosses all the remaining coins and removes the coins that come up tails. He repeats the previous step until there is one kind of coins remaining or there are no coins remaining. If there is one kind of coins remaining, then this kind of coins is lucky. Given the number of coins and the probability that the coins come up heads after tossing for each kind, your task is to calculate the probability for each kind of coins that will be lucky.

Input

The first line is the number of test cases. For each test case, the first line contains an integer k representing the number of kinds. Each of the following k lines describes a kind of coins, which contains an integer and a real number representing the number of coins and the probability that the coins come up heads after tossing. It is guaranteed that the number of kinds is no more than 10, the total number of coins is no more than 1000000, and the probabilities that the coins come up heads after tossing are between 0.4 and 0.6.

Output

For each test case, output a line containing k real numbers with the precision of 6 digits, which are the probabilities of each kind of coins that will be lucky.

Sample Input

```
3
1
1000000 0.5
2
1 0.4
1 0.6
3
2 0.4
2 0.5
2 0.6
```

Sample Output

```
1.000000
0.210526 0.473684
0.124867 0.234823 0.420066
```

Statistic | Submit | Clarifications | Back

Fibonacci

Time Limit: 8000/4000 MS (Java/Others) Memory Limit: 65536/65536 K (Java/Others)
Total Submission(s): 12 Accepted Submission(s): 1

Problem Description

We consider the Fibonacci sequence f where f(0) = 0, f(1) = 1 and f(n) = f(n - 1) + f(n - 2) for n ≥ 2. For given x(0), one can define another sequence x that x(n) = f(x(n-1)). Now you need to find the minimum n such that x(n) ≡ x(0) (mod p).

Input

The first line contains an integer T indicating the number of test cases. Then for each test case, a line consists of two integers x(0) and p where $0 \le x(0) \le 10^9$ and $1 \le p \le 200000$.

Output

For each test, output the minimum n in a line, or -1 if it is impossible.

Sample Input

```
5
6 4
8 11
9 11
12 11
13 11
```

Sample Output

```
3
3
-1
1
1
```

Hint

In the first case, x(0) = 6 ≡ 2 (mod 4), x(1) = f(6) = 8 ≡ 0 (mod 4) and x(2) = f(8) = 21 ≡ 1 (mod 4), and therefore x(3) = f(21) = 10946 ≡ 2 (mod 4).

Statistic | Submit | Clarifications | Back

Lambda Calculus

Time Limit: 2000/1000 MS (Java/Others) Memory Limit: 65536/65536 K (Java/Others)
Total Submission(s): 102 Accepted Submission(s): 1

Problem Description

The **lambda calculus** is a simple language that is often used to study the theory of programming languages. This language only consists of variable references, functions that take a single argument, and applications of functions.
The lambda calculus consists of a language of **lambda terms**. The syntax of the lambda calculus defines that all valid lambda terms can be built by applying the following three rules:
1. a **variable** x, which is a nonempty string other than lambda, is itself a lambda term;
2. if t is a valid lambda term, and x is a variable, then (lambda (x) t) is a lambda term (called a **lambda abstraction**), here x has at least one occurrence in t;
3. if t and s are lambda terms, then (t s) is a lambda term (called an **application**).
Nothing else is a lambda term. Thus a lambda term is valid if and only if it can be obtained by repeated application of these three rules. Note that the parentheses in the second and the third rule can NOT be omitted.
A **lambda abstraction** (lambda (x) t) is a definition of an anonymous function that is capable of taking a single input x and substituting it into the term t. The abstraction **binds** the variable x in term t, so we call it a **lambda binding** of variable x.
An **application** (t s) represents the application of a function t to an input s, that is, it represents the act of calling function t on input s to produce t(s).
To see how this works, consider the lambda calculus extended with arithmetic operators. In that language, a lambda abstraction (lambda (x) x + 5), in which x is bound, defines a function f(x) = x + 5. And an application ((lambda (x) x + 5) 3) applies function f(x) = x + 5 to input 3 and produces f(3) = 3 + 5 = 8.
We say that a variable **occurs free** in an lambda term t if it has some occurrence in t that is not inside some lambda binding of the same variable. For example,
● x occurs free in x;
● x does not occur free in y;
● x does not occur free in (lambda (x) (x y));
● x occurs free in (lambda (y) (x y));
● x occurs free in ((lambda (x) x) (x y)), which is an application that produces (x y) in which x occurs free;
● x occurs free in (lambda (y) (lambda (z) (x (y z)))).
Now you are given a lambda term t, you need to find all distinct variables that occur free in t.

Input

The first line of the input contains a positive integer T denoting the number of test cases. Then T lines follow, each line contains a nonempty string denoting a lambda term.
Each variable in the the lambda terms contains only latin letters and the hyphen, and is no longer than 20 characters.
It is guaranteed that all lambda terms in the input are valid, and the total length of all lambda terms will not exceed 107 .
There might be some extra spaces in the input as long as they do not cause any misunderstanding.

Output

For each test case, output first the case number and then all distinct variables that occur free in the lambda term in a single line. These variables should be printed in lexicographic order, and there should be a single space between
each two adjacent variables. There should be a space after the colon in each test case.

Sample Input

```
6
x
y
(lambda (x) (x y))
(lambda (y) (x y))
( (lambda (x) x) (x y) )
( lambda (y) ( lambda (z) (x (y z)) ) )
```

Sample Output

```
Case #1: x
Case #2: y
Case #3: y
Case #4: x
Case #5: x y
Case #6: x
```

Hint

The free variables of a term are those variables not bound by a lambda abstraction. The set of free variables of an expression is defined inductively:
1. The free variables of x are just x;
2. The set of free variables of (lambda (x) t) is the set of free variables of t, but with x removed;
3. The set of free variables of (t s) is the union

We can define it with the context-free grammar:

LcExp ::= Variable
::= (lambda (Variable) LcExp)
::= (LcExp LcExp)

where a variable is any string other than lambda.

Statistic | Submit | Clarifications | Back

Home | Top **Administration**

Coding Contest

Time Limit: 2000/1000 MS (Java/Others) Memory Limit: 65536/65536 K (Java/Others)
Total Submission(s): 257 Accepted Submission(s): 47

Problem Description

A coding contest will be held in this university, in a huge playground. The whole playground would be divided into N blocks, and there would be M directed paths linking these blocks. The i-th path goes from the u_i-th block to the v_i-th block. Your task is to solve the lunch issue. According to the arrangement, there are s_i competitors in the i-th block. Limited to the size of table, b_i bags of lunch including breads, sausages and milk would be put in the i-th block. As a result, some competitors need to move to another block to access lunch. However, the playground is temporary, as a result there would be so many wires on the path.
For the i-th path, the wires have been stabilized at first and the first competitor who walker through it would not break the wires. Since then, however, when a person go through the i - th path, there is a chance of p_i to touch
the wires and affect the whole networks. Moreover, to protect these wires, no more than c_i competitors are allowed to walk through the i-th path.
Now you need to find a way for all competitors to get their lunch, and minimize the possibility of network crashing.

Input

The first line of input contains an integer t which is the number of test cases. Then t test cases follow.
For each test case, the first line consists of two integers N ($N \le 100$) and M ($M \le 5000$). Each of the next N lines contains two integers si and b_i (s_i , $b_i \le 200$).
Each of the next M lines contains three integers u_i , v_i and c_i ($c_i \le 100$) and a float-point number p_i ($0 < p_i < 1$).
It is guaranteed that there is at least one way to let every competitor has lunch.

Output

For each turn of each case, output the minimum possibility that the networks would break down. Round it to 2 digits.

Sample Input

```
1
4 4
2 0
0 3
3 0
0 3
1 2 5 0.5
3 2 5 0.5
1 4 5 0.5
3 4 5 0.5
```

Sample Output

```
0.50
```

Statistic | Submit | Clarifications | Back

Pattern

Time Limit: 2000/1000 MS (Java/Others) Memory Limit: 65536/65536 K (Java/Others)
Total Submission(s): 0 Accepted Submission(s): 0

Problem Description

Alice and Bob are playing a game. Alice takes a paper with N × M points arranging in N rows and M columns.
First, Alice colors some points with one of 5 colors: c_0, c_1, c_2, c_3, c_4. And then, Bob draws some lines between adjacent points which own a common edge. If the color of a point is c_i , Bob must draw exactly i lines linking this point. Otherwise, Bob can draw any number of lines linking it. At last, Alice would color the rest points, with the same rules that the point which links i lines should be painted the color c_i . After the game, Alice might get different patterns of the colors. Suppose the initial colored paper can lead to totally K patterns, and there are P_i ways for Bob to draw lines for the i-th patterns. Alice wants to know $\sum_{i=1}^{K} P_i^2$.

Input

The first line of input contains an integer t which is the number of test cases. Then t test cases follow.
For each test case, the first line consists of two integers N($N \le 66$), and M($M \le 6$). The i-th line of the next N lines contains M integers, and among them the j-th integer donates the point (i, j). If the point is panted color c_i , the integer is i, otherwise it is -1.

Output

For each test cases, output $\sum_{i=1}^{K} P_i^2$ modulo 10007.

Sample Input

```
2
2 2
-1 -1
-1 -1
3 3
1 1 1
1 0 1
1 1 1
```

Sample Output

```
18
4
```

Hint

In the first case, there would be 15 patterns:

$$\begin{bmatrix}0&0\\0&0\end{bmatrix}\ \begin{bmatrix}0&0\\1&1\end{bmatrix}\ \begin{bmatrix}1&1\\0&0\end{bmatrix}\ \begin{bmatrix}0&1\\0&1\end{bmatrix}\ \begin{bmatrix}1&0\\1&0\end{bmatrix}$$

$$\begin{bmatrix}1&2\\0&1\end{bmatrix}\ \begin{bmatrix}2&1\\1&0\end{bmatrix}\ \begin{bmatrix}1&0\\2&1\end{bmatrix}\ \begin{bmatrix}0&1\\1&2\end{bmatrix}\ \begin{bmatrix}2&2\\2&2\end{bmatrix}$$

$$\begin{bmatrix}2&2\\1&1\end{bmatrix}\ \begin{bmatrix}1&1\\2&2\end{bmatrix}\ \begin{bmatrix}2&1\\2&1\end{bmatrix}\ \begin{bmatrix}1&2\\1&2\end{bmatrix}$$

There is only one way to draw lines which leads to above 14 patterns respectively.

$$\begin{bmatrix}1&1\\1&1\end{bmatrix}$$

There are two ways to draw lines which lead to above pattern. Therefore the answer is $18 = 14 + 2^2$.

Statistic | Submit | Clarifications | Back

Travel Brochure

Time Limit: 4000/2000 MS (Java/Others) Memory Limit: 32768/32768 K (Java/Others)
Total Submission(s): 2 Accepted Submission(s): 2

Problem Description

Welcome to Galilei Town, a high and new technology industrial development zone surrounding the Dishui lake. N villages numbered from 0 to N - 1 are located along the lake and a loop-line bus is the only transportation in this
town. The bus is a one-way line passing the villages 0, 1, 2, · · · , N - 1 successively, going back to the 0-th village and continuing the above route.
We may measure the landscape of the i-th villages by an integer w_i and $\sum_{i=0}^{N-1} w_i = 0$. Once a traveller takes the bus from the u-th village to the v-th village, he would evaluate the experience by two coefficients a = w_v and b the sum of w(s) for all passing villages (including the last stop and excluding the first one).
Now, as the tour guide, you need to design a travel brochure for guests who came from far away. Your task is to choose a village i_0 as the starting village of the travel and at least two more villages $i_1, i_2, \ldots, i_k$. Guests would
start their travel from the i_0-th village and visit the planned k villages in sequence by loop-line bus. Finally they will go back to the i_0-th village from the i_k-th one and finish their travel. If we let $i_k + 1 = i_0$, the whole travel would be evaluated by the score $\sum_{j=0}^{k}(a_{i_{j+1}} - a_{i_j}) \frac{b_{i_j} b_{i_{j+1}}}{a_{i_j} a_{i_{j+1}}}$. You need to know the maximum possible score

Input

The first line of input contains an integer t which is the number of test cases. Then t test cases follow. For each test case, the first line consists of an integer N ($3 \le N \le 100000$). The second line consists of N non-zero integers
w_0 to w_{N-1} where each w_i satisfies $|w_i| \le 100$. We guarantee that the sum of w_i would be zero.

Output

For each case, output the maximum score of the whole evaluation rounded to 5 decimal places behind the decimal point in a line.

Sample Input

```
1
10
1 4 1 2 -3 -5 2 -2 2 -2
```

Sample Output

```
28.66667
```

Hint

The best route starts from the 5-th village. Go passing the 6-th, 7-th, 8-th, 9-th, 0-th, 1-st, 2-nd, 3-rd villages and arrive at the 4-th village. Then go around the lake to the 3-rd villa

Cliques

Time Limit: 12000/6000 MS (Java/Others) Memory Limit: 65536/65536 K (Java/Others)
Total Submission(s): 2 Accepted Submission(s): 0

Problem Description

In graph theory, we call a subgraph of a undirected graph clique if each pair of nodes in it owns an edge between them. Consider a undirected graph G with n nodes. We hope to modify G into a new graph such that each connected component of it should be a clique. Each step of modification can be inserting a new edge or deleting an edge which has already existed.

Input

The first line of the input contains an integer t which is the number of test cases. For each test case, the first line consists of an integer n ($1 \le n \le 100$) which is the number of nodes in undirected graph G. Each of the following n lines consists of n integers corresponding to the adjacency matrix of G where 1 represents the existence of the edge and 0 otherwise.

Output

For each test case, output first the case number, see the sample output. There should be a space after the colon in each test case. Then, if the graph G can be modified into a new graph with no more than ten steps, output the minimum number of steps we need, or - 1 if not.

Sample Input

```
2
7
0 1 1 0 0 0 0
1 0 1 1 0 0 0
1 1 0 1 0 0 0
0 1 1 0 1 0 0
0 0 0 1 0 1 1
0 0 0 0 1 0 1
0 0 0 0 1 1 0
1
0
```

Sample Output

```
Case #1: 2
Case #2: 0
```

Statistic | Submit | Clarifications | Back

Finding Hotels

Time Limit: 2000/1000 MS (Java/Others) Memory Limit: 102400/102400 K (Java/Others)
Total Submission(s): 251 Accepted Submission(s): 9

Problem Description

There are N hotels all over the world. Each hotel has a location and a price. M guests want to find a hotel with an acceptable price and a minimum distance from their locations. The distances are measured in Euclidean metric.

Input

The first line is the number of test cases. For each test case, the first line contains two integers N ($N \le 200000$) and M ($M \le 20000$). Each of the following N lines describes a hotel with 3 integers x ($1 \le x \le N$), y ($1 \le y \le N$) and c ($1 \le c \le N$), in which x and y are the coordinates of the hotel, c is its price. It is guaranteed that each of the N hotels has distinct x, distinct y, and distinct c. Then each of the following M lines describes the query of a guest with 3 integers x ($1 \le x \le N$), y ($1 \le y \le N$) and c ($1 \le c \le N$), in which x and y are the coordinates of the guest, c is the maximum acceptable price of the guest.

Output

For each guests query, output the hotel that the price is acceptable and is nearest to the guests location. If there are multiple hotels with acceptable prices and minimum distances, output the first one.

Sample Input

```
2
3 3
1 1 1
3 2 3
2 3 2
2 2 1
2 2 2
2 2 3
5 5
1 4 4
2 1 2
4 5 3
5 2 1
3 3 5
3 3 1
3 3 2
3 3 3
3 3 4
3 3 5
```

Sample Output

```
1 1 1
2 3 2
3 2 3
5 2 1
2 1 2
2 1 2
1 4 4
3 3 5
```

Statistic | Submit | Clarifications | Back

Tower Attack

Time Limit: 8000/4000 MS (Java/Others) Memory Limit: 65536/65536 K (Java/Others)
Total Submission(s): 77 Accepted Submission(s): 1

Problem Description

There was a civil war between two factions in Skyrim, a province of the Empire on the continent of Tamriel. The Stormcloaks, led by Ulfric Stormcloak, are made up of Skyrim's native Nord race. Their goal is an independent Skyrim free from Imperial interference. The Imperial Legion, led by General Tullius, is the military of the Empire that opposes the Stormcloaks and seeks to reunite and pacify the province.
The current target of Ulfric Stormcloak is to attack Whiterun City, which is under controlled by General Tullius. Near by this city there are N towers under the Empire's control. There are N - 1 roads linking these towers, so soldiers can move from any tower to another one through these roads.
In military affairs, tactical depth means the longest path between two of all towers. Larger the tactical depth is, more stable these towers are.
Your mission, should you choose to accept it. In each turn, Ulfric tells you two roads. You need to figure out the tactical depth after destroying these two roads.

Input

The first line of input contains an integer t which is the number of test cases. Then t test cases follow. For each test case, the first line contains two integers N($N \leq 100000$), representing the number of towers, and Q($Q \leq 100000$), represeting the number of queries.
In the next N - 1 lines, the i-th line describes the i-th road and contains three integers u, v and w ($0 \leq w \leq 5000$) corressponding to a path between u and v of length w.
In the next Q lines, each line contains two integers u and v representing that the u-th road and the v-th road would be destroyed in this turn. It is guaranteed that $u \neq v$.

Output

For each query, output the tactical depth.

Sample Input

```
1
8 3
2 1 7
3 1 7
4 2 5
5 2 6
4 6 3
7 2 8
1 8 2
4 6
2 3
5 6
```

Sample Output

```
22
17
20
```

Statistic | Submit | Clarifications | Back

Generator and Monitor

Time Limit: 32000/16000 MS (Java/Others) Memory Limit: 70000/70000 K (Java/Others)
Total Submission(s): 0 Accepted Submission(s): 0

Problem Description

You have a generator that **randomly generates a uniformly distributed** random number in range [1, m] and a monitor that monitors the generator and gives an alert when some conditions are met. If at the i-th second, a condition is given to the monitor, then the monitor will give an alert at the j-th second reporting that the condition given at the i-th second is met. Here j is the minimum integer such that the numbers generated by the generator from the i-th second to the j-th second and in range [l_i, r_i] equals to c_i . Otherwise, the generator will generate a number a_i . However, the monitor stops working now. We need you to write a program to give the alerts.

Input

The first line contains an integer T, the number of test cases.
For each test case, the first line contains an integer m and n. The following n lines describe events. The i-th line describe the event happens at the i-th second. If the event is to give a condition, then it contains a character "C" followed by l_i, r_i and c_i . Otherwise, the event is to generate a number and it contains a character "G" followed by b_i . a_i , the number generated at the i-th second equals to the XOR sum of bi and the moments of all conditions met before i-th second.
It is guaranteed that $1 \le m \le 10^4$ and $1 \le n \le 2 \times 10^5$.

Output

For each test case, the output starts with a line "Case #i:" where i is the test case number, starting from 1. Then you need to report the alerts in order. If some conditions are met at the i-th second, output a line containing i and moments when these conditions were given. Output moments in increasing order.

Sample Input

```
1
6 5
C 1 3 1
C 3 5 2
G 4
G 1
G 3
G 2
```

Sample Output

```
Case #1:
4 1
6 2
```

Hint

At the 1st second, a condition is given. We need to make an alert when 1 number in range [1, 3] are generated.
At the 2nd second, a condition is given. We need to make an alert when 2 number in range [3, 5] are generated.
At the 3rd second, a number is generated. It is 4 because no condition is met before. We don't need to make any alert because no condition is met after 4 is generated.
At the 4th second, a number is generated. It is 1 because no condition is met before. We need to make an alert because condition at 1st second is met after 1 is generated.
At the 5th second, a number is generated. Because condition at the 1st second is met before, the actual number generated should be 3 xor 1 which is 2. We don't need to make any
At the 6th second, a number is generated. Because condition at the 1st second is met before, the actual number generated should be 2 xor 1 which is 3. We need to make an alert be

Statistic | Submit | Clarifications | Back