Problem A
Goalkeepers

When I was a high school student, we took the game of soccer very seriously. In our class team, there were two goalkeepers – one of them being myself. Being the GREAT fellow that I am, you certainly know who was the better goalkeeper. But for some very strange and unfair reasons the other goalkeeper somehow always got to play the games. But one day fate intervened. During the first half of the game, our other goalkeeper conceded 7 goals! During the half time interval, our classmates got very furious and angrily shouted to the coach that he must change the goalkeeper at second half and give the world class keeper at his bench (That's me! :D) a go. So, I played the second half as a substitute goalkeeper. Then guess what. As this is not a fairy tale, I put on an assured and accomplished display of goalkeeping and let in another 7 goals in our net. So, that is 7 in the first half and 7 in the second half – a total of 14 goals conceded in the match.

Our opponent got very fond of both the keepers and started a campaign to play both of us in the following matches, one in the first half and the other in the second. Now, you are given the number of goals conceded in the first half and in the second half of some matches, you'll have to tell me the total number of goals we conceded in the match, I am too embarrassed to count them after all. :)

Input

The 1st line of input contains only an integer, T (T<=100). Each of the following T lines contains 2 integers first and second – the number of goals conceded in the first half and in the second half respectively. You can assume that for all cases 0<=first, second <=100.

Output

For each test case, print a line in the format, “Case V: Y”, where V is the case number & Y is the total number of goals our team conceded in the match.

Sample Input

Output for Sample Input

<table>
<thead>
<tr>
<th>Sample Input</th>
<th>Output for Sample Input</th>
</tr>
</thead>
</table>
| 2
2 2
7 7          | Case 1: 4
             | Case 2: 14 |

NOTE: The following C / C++ code can be used for this problem:

```c
#include<stdio.h>

int main()
{
    int t,i,first,second,ans;
    scanf("%d",&t);
    for(i = 1; i <= t; i++)
```
{ 
    scanf("%d%d", &first, &second); 

    // write your code or formula here 
    // for example, if you think 
    // ans = first minus second then 
    // write "ans = first - second;" (without quotes) 

    // printing the answer 
    printf("Case %d: %d\n", i, ans); 
} 

return 0; 
}
Problem B
Beautiful Flower

Input: Standard Input
Output: Standard Output

A beautiful flower is floating on the lake which has its root in the soil under the water. This scenario is shown in the picture given below:

![Diagram of a flower floating on a lake with distances labeled]

Sometimes the flower moves to its right due to air flow and touches the water at point C. We know the distance \(d\) (BC) and the distance \(x\) (BE), which is the distance of the flower from water level when it is not affected by air i.e., AB & AE are perpendicular to BC. Given \(d\) and \(x\), we need to find the depth of the lake.

**Input**

The 1st line of input contains only an integer, \(T\) (\(T\leq 100\)). Each of the following \(T\) lines contains 2 integers \(x\) and \(d\). You can safely assume that \(d\) is always greater than \(x\) in any input case.

**Output**

For each test case, print a line in the format, “Case \(V\): \(Y\)”, where \(V\) is the case number & \(Y\) is the depth of the lake. Print 6 digits after the decimal point.

<table>
<thead>
<tr>
<th>Sample Input</th>
<th>Output for Sample Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 6 9 2 5</td>
<td>Case 1: 3.750000  Case 2: 5.250000</td>
</tr>
</tbody>
</table>

**NOTE:** The following C / C++ code can be used for this problem:

```c
#include<stdio.h>

int main()
{
    int t,i;
    double x,d,ans;
    scanf("%d",&t);
```
for(i = 1; i <= t; i++)
{
    scanf("%lf%lf", &x, &d);

    // write your code or formula here
    // for example, if you think
    // ans = x divided by d,
    // write "ans = x/d;" (without quotes)
    // if you think
    // ans = ( (x multiplied by d) - 2 ) divided by 3,
    // write “ans = (x*d - 2)/3;” (without quotes)
    // write your code or formula here

    // printing the answer
    printf("Case %d: %.6lf\n", i, ans);
}
return 0;
We have a very famous and popular fellow in our problem setters' panel. He is so famous that his name is immaterial. Some of his admirers have recently given him the nickname 'Emoogle'. Let's stick to that name in our discussion for now. Being such a kind, friendly and generous person as he is, Emoogle is often known to give treats to the other problem setters. Some times, there is a strange rumor in the air that his treats are mostly due to the fact that, if he is not sparing enough for those treats, 'problems' will be created. But let's not pay heed to such nonsense!

Now, in our gatherings, if the number of people present is divisible by 3 & 5 but not divisible by 4 then the smart Emoogle somehow manages a way to escape without giving a treat. So, we call the numbers that are divisible by 3 & 5 but not are not divisible by 4 to be Emoogle numbers. Given two integers, A & B, your task is to count the number of Emoogle numbers between A & B (inclusive)

**Input**

The 1st line of input contains only an integer, T (T<=100). Each of the following T lines contains 2 integers A & B (0<=A,B<=10000). You can also assume that A <= B.

**Output**

For each test case, print a line in the format, “Case V: Y”, where V is the case number & Y is the number of Emoogle numbers between A & B inclusively.

<table>
<thead>
<tr>
<th>Sample Input</th>
<th>Output for Sample Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 100 60</td>
<td>Case 1: 5</td>
</tr>
<tr>
<td></td>
<td>Case 2: 0</td>
</tr>
</tbody>
</table>
Problem D-E
Can You Swap It !!!!!

Input: Standard Input
Output: Standard Output

[Note: This problem has two versions. The easier version has smaller input range and this version is Problem D. On the other hand, the same problem only with a larger range is treated as Problem E. Please check the Input specification section for more detail about the input range for each problem.]

You are given a permutation of the numbers $1, 2, 3, \ldots, n$ (All the numbers from $1, 2, 3, \ldots, n$ are given but not in any particular order). At any single move you can swap (exchange) any two neighboring elements. If two numbers are next to each other they are neighbors. What is the minimum of number of swaps needed to sort them in ascending order?

Input

The input contain multiple test cases. Each case starts with a line containing a positive integer $n$. Next line contains $n$ space separated integers. All these integers are from 1 to $n$ and they are distinct. The last case contains $n$ where $n=0$. This case should not be processed.

For the easy version (Problem D) $1 \leq n \leq 1000$
For the hard version (Problem E) $1 \leq n \leq 100000$

Output

For every test case, on a single line output the minimum number of swap needed to make the list sorted in ascending order.

<table>
<thead>
<tr>
<th>Sample Input</th>
<th>Output for Sample Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 1 3 2 3 3 3 2 1 0</td>
<td>1 3</td>
</tr>
</tbody>
</table>

Output Explanation:
In the second test case first two elements are swapped. Now it becomes 2 3 1. Then last two elements are swapped, It becomes 2 1 3. Then finally first two elements are again swapped. So it becomes 1 2 3.
Problem F-G
Bitwise Mania

Input: Standard Input
Output: Standard Output

[Note: This problem has two versions. The easier version has smaller input range and this version is Problem F. On the other hand, the same problem only with a larger range is treated as Problem G. Please check the Input specification section for more detail about the input range for each problem.]

A bitwise operation operates on one or more bit patterns at the level of their individual bits. Being learned about AND, OR, XOR and NOT these four bitwise operators, your cryptography course advisor have given you some challenging task to convert a initially given 32 bit unsigned number(U) to another 32 bit unsigned number(V) using a Key value. Unfortunately, he’s forgotten to tell you the Key, so you have to use U as Key. Now, at each conversion, you have four options to operate with the number:

* Bitwise AND with U
* Bitwise OR with U
* Bitwise XOR with U
* Bitwise NOT

But, the fact is you must use all of these four bitwise operators at least one time and total bitwise operation must be minimum M times. That is, if you use total a time AND operator, b time OR operator, c time XOR operator and d time NOT operator, then these two conditions must suffice:

* a, b, c, d all must be non-zero
* a + b + c + d >= M

Obviously , you can use any operations in any order .

So, given U, V, M, you have to find out is it possible to convert U to V using minimum M bitwise conversion. If it possible, then find out minimum bitwise conversion ( >= M ) needed to convert U to V.

Input

There will be at most 1007 test cases in the input file. Each case contains 3 unsigned integer U (0<=U<2^32), V (0<=V<2^32) and M (M >=4) in a line.

For easy version (Problem F), M <=10
For hard version, (Problem G) M<=1000007

The end of input will be denoted by a case with U = 0, V = 0, M = 0. This case should not be processed.
Output

For each test case, if it is possible to convert, then you have to print a line in the format, “Case X: Y”, where X is the case number & Y is minimum bitwise conversion ( \( \geq M \) ) needed to convert U to V. Otherwise, print “Case X: -1”.

<table>
<thead>
<tr>
<th>Sample Input</th>
<th>Output for Sample Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 4</td>
<td>Case 1: 4</td>
</tr>
<tr>
<td>23 45 7</td>
<td>Case 2: -1</td>
</tr>
<tr>
<td>67 4294967295 6</td>
<td>Case 3: 6</td>
</tr>
</tbody>
</table>

Explanation of sample input:

In the first sample, you can do the following conversion:

0 OR 0 => 0
NOT 0 => 4294967295
4294967295 AND 0 => 0
0 XOR 0 => 0

So, in 4 conversion using all the four bitwise operator 0 can be converted to 0 using key 0.

In the second sample, you can’t do the conversion from 23 to 45 using key 23 anyhow you try.

But in the third sample you can do the following conversion:

67 AND 67 => 67
67 XOR 67 => 0
0 OR 67 => 67
67 AND 67 => 67
NOT 67 => 4294967228
4294967228 OR 67 => 4294967295

Total 6 operations. You may check out that you can do the same conversion in 5 or 9 operation, but as M is 6, so you have to find minimum conversion greater than or equal 6.

So, output will be 6.

Truth table for the bitwise operations:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>NOT A</th>
<th>A AND B</th>
<th>A OR B</th>
<th>A XOR B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Problem H-I
Birthday Party

Input: Standard Input
Output: Standard Output

[Note: This problem has two versions. The easier version has smaller input range and this version is Problem H. On the other hand, the same problem only with a larger range is treated as Problem I. Please check the Input specification section for more detail about the input range for each problem.]

Little Sushi is going to celebrate her tenth birthday soon. It’s a very exciting moment for her. She can’t believe that she has spent a decade in this world. That’s amazing, isn’t it! She would like to make this birthday to be remembered by all of her friends. So she came out of an idea.

There is a candy shop in her city named “Candies for You”. You can only buy candies there, nothing else! But they have a large collection of candy boxes. All the boxes are arranged in a row one by one. Interestingly, each of the box costs the same, and it’s 100 taka per box only. But you know, unfortunately, Little Sushi doesn’t have a lot of money in her small bank. She found that she just has enough money to buy 2(two) candy boxes only. Poor Sushi!

But life is not that easy. Going to the shop, she finds that all the boxes do not have the same number of candies(though the price is same). So she wants to buy two candy boxes such that she can distribute all the candies to her friends evenly(i.e. each friend gets the same number of candies and no candy is left out). For example, let she has 5 friends, and she bought two boxes with 8 and 7 candies. So she has in total 8 + 7 = 15 candies, which she can distribute to her friends evenly, each of them getting 3 candies(as 15/5 = 3). But if she buys two boxes with 8 and 8 candies(boxes are different, but they have same number of candies), then she will have 8 + 8 = 16 candies in total. And giving 3 candies to each of her friends will distribute 15 candies, and 1 candy will remain. So this is not a valid buying option for her.

Another problem is that, the owner of the shop replaces candy box when some new boxes arrive. And that can happen when Sushi is inside the shop, looking for boxes to buy. So she needs your help. She will ask you such questions that how many ways she can buy candies satisfying her condition if she decides to buy boxes from ranges i to j (inclusive).

Input

The first line of the input contains an integer T, the number of test cases. Each case will start with three integer, F (1 <= F <= 20), number of friends Sushi has, N (see constraints below), total number of candy boxes, and M, number of queries. Each of the next N lines will contain an integer representing the number of candies in boxes. Each of the next M lines will contain 3 integers, type, a, b. type (1 for updating candy box, 2 for questions that Sushi ask). For update query, later two number(a, b) will be
the position(1-based) of candy box which is replaced, and the number of candies in the new box. For questions of Sushi those two numbers(a and b) will be the positions between which Sushi would like to search for option. Positions are 1-based and will be between 1 and N, and you have to search inclusive i.e. both positions are also included in search. And you can safely assume that a <= b in second type.

- You have to take two different boxes.
- Number of candy in each box at any time will be between 1 and 1,000,000,000 (inclusive).
- For every type 2 query, i.e. 2 a b, a <= b.
- 1 <= F <= 20

For easy version, (Problem H), 1 <= T <= 10, 1 <= N <= 1000, 1 <= M <= 50
For hard version, (Problem I), 1 <= T <= 3, 1 <= N <= 40000, 1 <= M <= 40000

Output

For each case, the output will start with a line “Case #<case_no>:” in a single line without the quotes(“) and case_no representing the case number for that case starting with 1. Each of the next output lines for that case is the questions asked by Sushi in queries (i.e. queries of type 2).
For each question that Sushi ask, output a number representing the number of options Sushi has. The number will fit in a 32-bit signed integer.

<table>
<thead>
<tr>
<th>Sample Input</th>
<th>Output for Sample Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 5 10 10 1 2 103 4 5 100 1 4 3 29 2 2 9 2 1 5 2 1 2 2 1 4 1 5 6 2 2 9 1 3 5 2 1 5 1 8 1305009 2 1 10</td>
<td>Case #1: 5 2 0 2 6 2</td>
</tr>
</tbody>
</table>
Problem J-K
Pole and Wire
Input: Standard Input
Output: Standard Output

[Note: This problem has two versions. The easier version has smaller input range and this version is Problem J. On the other hand, the same problem only with a larger range is treated as Problem K. Please check the Input specification section for more detail about the input range for each problem.]

There are \( N \) poles along a road side. You have to tie wires to the poles. You can attach two ends of a wire to two different poles. A pole might not have any wire attached to it. A pole also must not have more than one wire attached to it. Two wires must not cross each other but may overlap. For example, suppose there are 5 poles. Say a wire is from pole 1 to pole 4 and another from pole 2 to pole 5. These two wires crossed each other. So such configuration is invalid. But say a wire is from pole 1 to pole 4 and another from pole 2 to pole 3. This is overlap, so it is valid configuration.

Input

First line of the test file contains number of test cases, \( T(\leq 300) \). For every test case you will be given number of poles \( N \),

For the easy version (Problem J) \( 1 \leq N \leq 50 \)
For the hard version (Problem K) \( 1 \leq N \leq 10^5 \)

Output

For every test case you are to output number of test case and number of ways of valid configuration. As the number of ways is very big you have to output the result in modulo 1000003.

<table>
<thead>
<tr>
<th>Sample Input</th>
<th>Output for Sample Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Case 1: 4</td>
</tr>
<tr>
<td>3</td>
<td>Case 2: 9</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Hint:

For \( N = 3 \), there are 4 possible valid configurations.

* No wires attached
* Wire from Pole 1 to Pole 2
* Wire from Pole 2 to Pole 3
* Wire from Pole 1 to Pole 3