The 2025 Asia Nanjing Regional Contest

Contest Session

November 9, 2025

Problem List

A	Wow, It’s Yesterday Six Times More
B	What, More Kangaroos?
C	Distributing Candies
D	Fallleaves01 and Golf
E	Cyan White Tree
F	Bitwise And Path
G	Bucket Bonanza
H	Pen Pineapple Apple Pen
I	Chi Fan
J	Trajan Algorithm
K	Xiangqi
L	Regional Champion
M	Many Convex Polygons

This problem set should contain 13 (thirteen) problems on 20 (twenty) numbered pages. Please inform a runner immediately if something is missing from your problem set.

Prepared by SUA Programming Contest Problem Setter Team.
https://sua.ac/

Hosted by

Problem Set Prepared by

It’s against the rules to open non-contest websites during the contest.
If you’re interested (which is our pleasure),
please scan the QR code only after the contest.

Problem A. Wow, It's Yesterday Six Times More

This is an interactive problem.

After the great success from 2018 to 2024, the Nanjing University of Aeronautics and Astronautics (NUAA) will host the *International Collegiate Programming Contest* (ICPC) Nanjing regional for the eighth time in a row.

Team ***Power of Two*** and team ***Three Hold Two*** won the champion title for Tsinghua University in 2018 and 2019. In 2020, 2021, and 2022, team ***Inverted Cross*** from Peking University won the three-peat champion titles, while in 2023, another team ***Reborn as a Vegetable Dog*** from Peking University won the title. In 2024, team ***Afterlife*** from Zhejiang University took the first place.

This year, around 335 teams are participating in the contest. At most 33 gold medals, 66 silver medals, and 99 bronze medals will be awarded (note that these numbers are for reference only). We are looking forward to seeing the participants' outstanding performance! We also want to express our gratitude for the hard work done by all staff and volunteers for this contest. Thank you all for your great contribution to this contest!

Photo taken in the 2023 ICPC Asia Nanjing Regional Contest

In the 2018 contest, problem K, ***Kangaroo Puzzle***, requires the contestants to construct an operation sequence for the game:

> The puzzle is a grid with n rows and m columns ($1 \le n, m \le 20$), and there are some (at least 2) kangaroos standing in the puzzle. The player's goal is to control them to get together. There are some walls in some cells, and the kangaroos cannot enter the cells with walls. The other cells are empty. The kangaroos can move from an empty cell to an adjacent empty cell in four directions: up, down, left, and right.
>
> There is exactly one kangaroo in every empty cell in the beginning, and the player can control the kangaroos by pressing the buttons U, D, L, R on the keyboard. The kangaroos will move simultaneously according to the button you press.
>
> The contestant needs to construct an operating sequence of at most 5×10^4 steps consisting of U, D, L, R only to achieve the goal.

In the 2020 contest, problem A, ***Ah, It's Yesterday Once More***, requires the contestants to construct an input map to hack the following code of the problem described before:

```
#include <bits/stdc++.h>
using namespace std;
string s = "UDLR";
int main()
{
  srand(time(NULL));
  for (int i = 1; i <= 50000; i++) putchar(s[rand() % 4]);
  return 0;
}
```

Furthermore, in the 2021 contest (Problem A, ***Oops, It's Yesterday Twice More***), the 2022 contest (Problem A, ***Stop, Yesterday Please No More***), the 2023 contest (Problem A, ***Cool, It's Yesterday Four Times More***), and the 2024 contest (Problem A, ***Hey, Have You Seen My Kangaroo?***), every year we have a problem related to the kangaroos! We would like to introduce all these problems to you, but if we do so every year, we may have a 500-page statement for one single problem in the 3025 contest. Therefore, we omit them this time. Besides, you may already have seen them in the practice contest.

Now, in the 2025 contest, as everyone expects, the kangaroo problem is back again! We don't know why problem setters are so obsessed with kangaroos, but the problem is as follows:

You are given a grid with n rows and m columns. There is a hole in the cell on the i_h-th row and the j_h-th column. All other cells are empty and there is one kangaroo standing in each cell.

Similarly, the kangaroos are controlled by pressing the button U, D, L, R on the keyboard. All kangaroos will move simultaneously according to the button pressed. Specifically, for any kangaroo located in the cell on the i-th row and the j-th column, indicated by (i, j):

1. Button U: it will move to $(i - 1, j)$.
2. Button D: it will move to $(i + 1, j)$.
3. Button L: it will move to $(i, j - 1)$.
4. Button R: it will move to $(i, j + 1)$.

If a kangaroo steps onto the hole (that is, $i = i_h$ and $j = j_h$) or steps out of the grid, it will be removed from the grid.

The problem is that, the exact value of i_h and j_h is not known. Your task is to find the position of the hole with at most $(n + m + 10)$ queries. For each query, you can press one button (U, D, L, or R). The interactor will output an integer t as the answer, indicating the number of kangaroos remaining after you press the button.

Note that the interactor is not adaptive, meaning that the answer for each test case is pre-determined.

Input

There are multiple test cases. The first line of the input contains an integer T $(1 \le T \le 100)$ indicating the number of test cases. For each test case:

The first line contains two integers n and m $(3 \le n, m \le 30)$ indicating the number of rows and columns in the grid.

Interaction Protocol

To ask a query, output one line. First output `?` followed by a space, then print one upper-case English letter (U, D, L, or R). After flushing your output, your program should read a single integer t indicating the answer to your query. Recall that you can ask at most $(n + m + 10)$ queries for a test case.

If you want to guess the position of the hole, output one line. First output `!` followed by a space, then print two integers i_h and j_h $(1 \le i_h \le n, 1 \le j_h \le m)$ separated by a space indicating the position. After flushing your output, your program should continue processing the next test case, or exit immediately if there are no more test cases. Note that your guess does not count as a query.

To flush your output, you can use:

- `fflush(stdout)` (if you use `printf`) or `cout.flush()` (if you use `cout`) in C and C++.
- `System.out.flush()` in Java and Kotlin.
- `sys.stdout.flush()` in Python.

Example

standard input	standard output
2	
3 4	
	? L
7	
	? D
3	
	! 2 3
4 3	
	? U
8	
	? R
5	
	! 1 1

Problem B. What, More Kangaroos?

There are n kangaroos on a number line, where the i-th kangaroo is initially at the positive coordinate c_i (that is, $c_i > 0$), and has two step lengths a_i and b_i.

The kangaroos are controlled by pressing the button 1, 2, 3 and 4 on the keyboard. All kangaroos will move simultaneously according to the button pressed. Specifically, let x_i be the current coordinate of the i-th kangaroo:

- Button 1: It will move to $(x_i + a_i)$.
- Button 2: It will move to $(x_i - a_i)$.
- Button 3: It will move to $(x_i + b_i)$.
- Button 4: It will move to $(x_i - b_i)$.

It is allowed for two or more kangaroos to be at the same coordinate. If two kangaroos meet during the movement, they will simply pass through each other.

You can press the buttons as many times as you want. Maximize the number of kangaroos with a negative coordinate (that is, maximize the number of kangaroos with $x_i < 0$) after the operations.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 10^5$) indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 2 \times 10^5$) indicating the number of kangaroos.

For the following n lines, the i-th line contains three integers a_i, b_i and c_i ($-10^9 \le a_i, b_i \le 10^9$, $1 \le c_i \le 10^9$) describing the i-th kangaroo.

It is guaranteed that the sum of n of all test cases does not exceed 2×10^5.

Output

For each test case, output one line containing one integer, indicating the maximum number of kangaroos with a negative coordinate after the operations.

Example

standard input	standard output

2
4
1 1 1
-1 3 4
0 -2 7
-5 0 10
1
0 0 1
	3
	0

Problem C. Distributing Candies

BaoBao and DreamGrid are good friends who often share candies together. Now BaoBao has a candies, and DreamGrid has b candies, making a total of $n = a + b$ candies.

What's more, BaoBao finds that these n candies can be divided into a piles with the same amount of candies, while DreamGrid finds that these n candies can also be divided into b piles with the same amount of candies.

After mixing the candies together, they forget how many candies each of them had. You need to help them find a possible pair of positive integers a and b, or indicate that it is impossible.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 10^5$), indicating the number of test cases. For each test case:

The first and only line contains an integer n ($1 \le n \le 10^{18}$), indicating the total number of candies.

Output

For each test case:

- If there exist such positive integers a and b, first output `Yes` in one line, then output two integers a and b separated by a space in another line. If there are multiple valid answers, you can output any of them.
- Otherwise, if there do not exist such positive integers a and b, just output `No` in one line.

Example

standard input	standard output
3 1 6 7	No Yes 3 3 No

Problem D. Fallleaves01 and Golf

Fallleaves01 has a deep passion for golf, but unfortunately, his skills are rather average.

To improve his game, Fallleaves01 purchased k indoor golf putting trainers for practicing at home. Unlike ordinary trainers, the ones he bought are very large. Specifically, each trainer can be abstracted as a tree with n_i vertices, where vertex 1 represents the hole. There is a golf ball on each trainer. For the i-th trainer, the ball is initially placed at vertex s_i.

During each stroke, Fallleaves01 chooses one trainer to play. Due to his limited skill level, the golf ball will move uniformly at random to one of the adjacent nodes. If the ball reaches the hole (vertex 1), the practice session ends for the day. Otherwise, Fallleaves01 repeats the above process until any ball enters a hole.

Fallleaves01's goal is to minimize the expected number of strokes required to finish the practice session. Calculate the expected number of strokes, assuming he always makes the best choice for each stroke.

Input

There is only one test case in each test file.

The first line contains an integer k ($1 \le k \le 300$), indicating the number of indoor golf putting trainers.

Then, the descriptions for the trainers follow. For each trainer:

- The first line contains two integers n_i and s_i ($2 \le s_i \le n_i \le 300$).
- For the following $(n_i - 1)$ lines, the j-th line contains two integers $u_{i,j}$ and $v_{i,j}$ ($1 \le u_{i,j}, v_{i,j} \le n_i$), indicating that there is an edge connecting vertices $u_{i,j}$ and $v_{i,j}$. It is guaranteed that these edges form a tree.

Output

Output one line containing one number, indicating the expected number of strokes under the optimal strategy.

Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}. Formally speaking, suppose that your output is a and the jury's answer is b, your output is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Example

standard input	standard output

2
4 2
1 2
2 3
3 4
4 3
1 2
2 3
3 4
	4.666666666666666

Problem E. Cyan White Tree

Given a tree with n vertices, each vertex is colored either cyan or white. For a simple path in the tree, let c be the number of cyan vertices and w be the number of white vertices in the path. The value of the path is $(c + w) - 2 \times |c - w|$, where $|c - w|$ is the absolute value of $(c - w)$.

Define the depth of a vertex u to be the number of edges on the shortest path from vertex 1 to vertex u. Define the key vertex of a simple path to be the vertex with minimum depth among all vertices in the path. It can be proven that the key vertex for a simple path is unique.

For each $1 \le i \le n$, calculate the maximum value among all simple paths whose key vertex is i.

Recall that a path in the tree is a nonempty sequence of vertices $p_1, p_2, \ldots, p_k$ such that each pair of adjacent vertices in this sequence is connected by an edge. A simple path is a path where all vertices in it are distinct. A single vertex can also form a simple path.

Input

There are multiple test cases. The first line of the input contains an integer T $(1 \le T \le 10^5)$ indicating the number of test cases. For each test case:

The first line contains an integer n $(2 \le n \le 4 \times 10^5)$ indicating the number of vertices.

The second line contains a `01`-string of length n, indicating the color of each vertex. If the i-th character is '`0`', the i-th vertex is colored cyan; otherwise, the i-th vertex is colored white.

For the following $(n - 1)$ lines, the i-th line contains two integers u_i and v_i $(1 \le u_i, v_i \le n)$, indicatng an edge connecting vertices u_i and v_i. It's guaranteed that the given edges form a tree.

It is guaranteed that the sum of n of all test cases does not exceed 4×10^5.

Output

For each test case output n lines, each containing one integer. The i-th line is the maximum value among all simple paths whose key vertex is i.

Example

standard input	standard output

3
8
00101110
1 2
1 3
3 4
3 5
5 6
5 7
7 8
2
01
1 2
2
00
1 2
	6
	-1
	3
	-1
	1
	-1
	2
	-1
	2
	-1
	-1
	-1

Problem F. Bitwise And Path

Given an undirected graph with n vertices, initially the graph has no edges. Maintain q operations of the following two types:

- $+$ u v w: Add an edge connecting vertices u and v, with a weight of w.
- $?$ u v: Define the weight of a path to be the bitwise-and of the weights of all edges in the path. Calculate the largest possible weight of a path starting from vertex u and ending at vertex v. If there is no path connecting vertices u and v, the answer should be -1.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 100$) indicating the number of test cases. For each test case:

The first line contains two integers n and q ($2 \le n \le 10^3$, $1 \le q \le 10^6$) indicating the number of vertices and the number of operations.

For the following q lines, the i-th line first contains a character op ($op \in \{+, ?\}$) indicating the type of the i-th query.

- If $op = +$, then three integers u, v and w follow ($1 \le u, v \le n$, $0 \le w < 2^{12}$), indicating an operation of the first type.
- If $op = ?$, then two integers u and v follow ($1 \le u, v \le n$, $u \ne v$), indicating an operation of the second type.

It's guaranteed that there is at least one operation of the second type for each test case. Also the sum of n of all test cases does not exceed 10^3, and the sum of q of all test cases does not exceed 10^6.

Output

To decrease the size of output, for each test case just output one line containing one integer, indicating the sum of the answers to all operations of the second type.

Example

standard input	standard output
2 4 11 + 1 3 6 + 3 4 2 ? 1 4 + 1 4 3 ? 4 1 + 4 3 4 ? 1 4 + 3 2 3 ? 1 2 + 1 1 4 ? 1 3 4 4 + 1 2 64 + 3 4 32 ? 1 2 ? 3 1	17 63

Note

For the first sample test case, the answers to the operations of the second type are 2, 3, 4, 2, and 6.

For the second sample test case, the answers to the operations of the second type are 64 and -1.

Problem G. Bucket Bonanza

Welcome to Mr. Ant Tenna's TV Time! In this round of game, you're given n buckets, where the i-th bucket has a maximum capacity of v_i volumes of water. At the start of the game, all buckets will be fully filled with water, and your mission is to keep it contained as much as you can for the next t seconds. But here's the twist: these buckets are broken! The i-th bucket has a leaking speed of l_i, meaning that every second after the game starts, the i-th bucket dribbles away l_i volumes of water.

Before the game starts, you may merge any number of buckets into one. The resulting bucket has a capacity equal to the maximum capacity among the merged buckets, and a leaking speed equal to the minimum leaking speed among the merged buckets. You can do the merging an arbitrary number of times, but they must be performed before filling any water.

Tenna poses q questions, where the i-th question has a time limit t_i. For each $1 \le i \le q$, calculate the maximum total retained volume after t_i seconds. Note that each question is independent.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 5 \times 10^4$) indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 2 \times 10^5$), indicating the number of buckets.

The second line contains n integers $v_1, v_2, \cdots, v_n$ ($1 \le v_i \le 4 \times 10^{13}$), where v_i is the capacity of the i-th bucket.

The third line contains n integers $l_1, l_2, \cdots, l_n$ ($0 \le l_i \le 10^9$), where l_i is the leaking speed of the i-th bucket.

The fourth line contains an integer q ($1 \le q \le 2 \times 10^5$), indicating the number of questions.

The fifth line contains q integers $t_1, t_2, \cdots, t_q$ ($0 \le t_i \le 10^9$), where t_i is the time limit for the i-th query.

It's guaranteed that neither the sum of n nor the sum of q of all test cases will exceed 2×10^5.

Output

For each test case, output one line containing q integers separated by a space, where the i-th integer is the maximum total retained volume after t_i seconds.

Example

standard input	standard output

2
4
5 4 7 6
2 1 3 2
3
3 1 2
4
19 47 21 13
5 14 2 3
5
5 2 6 1 4
	4 14 8
	43 67 38 77 48

Problem H. Pen Pineapple Apple Pen

In the spirit of the viral hit Pen-Pineapple-Apple-Pen (PPAP) by PIKOTARO, you need to find the number of ways to select four non-overlapping substrings from left to right in the given string $S = s_1 s_2 \dots s_n$ consisting only of lowercase English letters, so that the four substrings mimic the iconic PPAP structure:

pen pineapple apple pen

More specifically, choosing eight integers $i_1, j_1, i_2, j_2, i_3, j_3, i_4, j_4$ ($1 \le i_1 \le j_1 < i_2 \le j_2 < i_3 \le j_3 < i_4 \le j_4 \le n$), such that:

- The first and fourth substrings are identical: $s_{i_1} s_{i_1+1} \dots s_{j_1} = s_{i_4} s_{i_4+1} \dots s_{j_4}$. This represents the matching "pen" at the beginning and end.
- The third substring is a strict suffix of the second substring. That is, there exists an integer k_2 ($i_2 < k_2 \le j_2$) such that $s_{k_2} s_{k_2+1} \dots s_{j_2} = s_{i_3} s_{i_3+1} \dots s_{j_3}$. This captures the relationship between "pineapple" and "apple", where the latter is a strict suffix of the former.

You need to find the number of valid 8-tuples $(i_1, j_1, i_2, j_2, i_3, j_3, i_4, j_4)$ satisfying the above conditions.

Input

There is only one test case in each test file.

The first and only line contains a string S ($1 \le |S| \le 5 \times 10^3$) indicating the given string consisting only of lowercase English letters.

Output

Output one line containing one integer, indicating the number of valid PPAP-style selections. As the answer might be large, output it modulo 998 244 353.

Examples

standard input	standard output
ppaap	1
ppapppap	16
penpineappleapplepen	1502

Problem I. Chi Fan

Grammy has two teammates A and B. Usually, the two teammates go out to eat at nice restaurants every day, while Grammy generally eats at the school cafeteria. Eating at the school cafeteria saves time and money, while going out sometimes incurs taxi fares. Even so, there are times when Grammy chooses to go out with her teammates to improve her meals.

As the end of the month approaches, Grammy has only m yuan left for her living expenses, and she needs to pass the next n days with this money. On the i-th day, she has two choices:

- Eat at the cafeteria, gaining a satisfaction of a_i and spending b_i yuan.
- Go out to eat with her teammates, gaining a satisfaction of c_i and spending d_i yuan.

Since A and B go out to eat every day, there will be taxi fares each day. Let e_i be the taxi fare on the i-th day, the three of them have established a rule for paying the taxi fare:

- If Grammy does not go out to eat on the i-th day, then there is a $\frac{p_i}{100}$ probability that A will pay for the taxi fare, and a $(1 - \frac{p_i}{100})$ probability that B will pay.
- If Grammy goes out to eat on the i-th day, and since the last time Grammy paid for the taxi fare (or since the first day, if Grammy has never paid for the taxi fare), both A and B have each paid for the taxi fare at least once (regardless of whether Grammy went out to eat at that time), then Grammy will pay for the taxi fare this time. Otherwise, one of the teammates will pay according to the previous rule.

Each day, Grammy will choose her action before the meal. She will know who paid the taxi fare on the previous day, and her goal is to maximize the expected total satisfaction, while ensuring that the total expenses do not exceed m yuan in the worst-case scenario. If she makes optimal decisions every day, what is the expected value of the total satisfaction?

Input

There is only one test case in each test file.

The first line contains two integers n and m ($1 \le n \le 2 \times 10^3$, $1 \le m \le 5 \times 10^3$), indicating the number of days and the total expense limit.

For the following n lines, the i-th line contains six integers a_i, b_i, c_i, d_i, e_i, and p_i ($0 \le b_i, d_i, e_i \le 5 \times 10^3$, $0 \le a_i, c_i \le 10^9$, $0 \le p_i \le 100$). Their meanings are described above.

Output

Output one line containing one single number, indicating the maximum expected value of total satisfaction. If there does not exist a plan where the total expenses will be less than or equal to m yuan in the worst-case scenario, output -1.

Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}. Formally speaking, suppose that your output is a and the jury's answer is b, your output is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Examples

standard input	standard output

2 10
6 4 15 8 10 50
5 2 13 5 8 50
	20.000000000000

5 12
2 0 10 5 3 0
2 0 9 4 3 100
2 0 12 7 3 0
2 0 8 4 3 100
2 0 10 6 3 0
	25.000000000000

6 20
1 2 3 10 1 69
5 1 12 2 2 23
0 4 2 3 5 41
6 1 6 9 0 84
1 1 14 9 5 98
3 2 13 2 4 43
	35.771706880000

3 10
10 4 30 8 1 0
10 4 20 5 1 100
10 6 5 2 1 50
	-1

Problem J. Trajan Algorithm

Recall the *Tarjan Algorithm* for calculating the articulation points (also called the cut vertices) in an undirected graph, whose pseudo-code is shown on the left.

Algorithm 1 The Correct Tarjan Algorithm

```
 1: function TARJAN($i$, $d$)
 2:     $visited[i] \leftarrow$ true
 3:     $depth[i] \leftarrow d$
 4:     $low[i] \leftarrow d$
 5:     $childCount \leftarrow 0$
 6:     $isArticulation \leftarrow$ false
 7:
 8:     for each $j$ in $adj[i]$ do
 9:         if $j = parent[i]$ then
10:             continue
11:         end if
12:         if not $visited[j]$ then
13:             $parent[j] \leftarrow i$
14:             TARJAN($j$, $d + 1$)
15:             $childCount \leftarrow childCount + 1$
16:             if $low[j] \geq depth[i]$ then
17:                 $isArticulation \leftarrow$ true
18:             end if
19:             $low[i] \leftarrow \min(low[i], low[j])$
20:         else
21:             $low[i] \leftarrow \min(low[i], depth[j])$
22:         end if
23:     end for
24:     if ($parent[i] \neq$ null and $isArticulation$) or ($parent[i] =$ null and $childCount > 1$) then
25:         Output $i$ as articulation point
26:     end if
27: end function
```

Algorithm 2 The Incorrect Trajan Algorithm

```
 1: function TRAJAN($i$, $d$)
 2:     $visited[i] \leftarrow$ true
 3:     $depth[i] \leftarrow d$
 4:     $low[i] \leftarrow d$
 5:     $childCount \leftarrow 0$
 6:     $isArticulation \leftarrow$ false
 7:
 8:     for each $j$ in $adj[i]$ do
 9:         if $j = parent[i]$ then
10:             continue
11:         end if
12:         if not $visited[j]$ then
13:             $parent[j] \leftarrow i$
14:             TRAJAN($j$, $d + 1$)
15:             $childCount \leftarrow childCount + 1$
16:             if $low[j] \geq depth[i]$ then
17:                 $isArticulation \leftarrow$ true
18:             end if
19:         end if
20:         $low[i] \leftarrow \min(low[i], low[j])$
21:
22:
23:     end for
24:     if ($parent[i] \neq$ null and $isArticulation$) or ($parent[i] =$ null and $childCount > 1$) then
25:         Output $i$ as articulation point
26:     end if
27: end function
```

Note that in the pseudo code, $adj[i]$ is the array containing all vertices adjacent to vertex i, $parent$ is an array in which the initial value of each element is null, and $visited$ is an array in which the initial value of each element is false.

Some students may have a hard time understanding this algorithm, especially from line 19 to line 22. They can't understand why the *Tarjan Algorithm* needs to calculate the value of $low[i]$ differently, and they may bear an incorrect version of the algorithm in mind. Let's call this incorrect version of the algorithm the *Trajan Algorithm*, which is shown on the right. Notice that the *Tarjan* and the *Trajan* algorithms are almost the same, except for the function names and the 4 lines from line 19 to line 22.

Given an undirected connected graph with n vertices and m edges, for each vertex u determine if there exists a vertex v and a 2-dimensional array adj (adj must represent the given graph) such that TARJAN(v, 0) and TRAJAN(v, 0) produce different results on vertex u (that is to say, the *Tarjan Algorithm* states that vertex u is an articulation point while the *Trajan Algorithm* does not, or vice versa).

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \leq T \leq 10^4$) indicating the number of test cases. For each test case:

The first line contains two integers n and m ($2 \le n \le 10^5$, $1 \le m \le 10^5$) indicating the number of vertices and edges in the given graph.

For the following m lines, the i-th line contains two integers u_i and v_i ($1 \le u_i, v_i \le n$) indicating that there is an edge connecting vertex u_i and v_i in the graph.

It's guaranteed that the given graph is connected and there are no self loops or multiple edges. Also, neither the sum of n nor the sum of m of all test cases will exceed 10^6.

Output

For each test case output one line containing some integers separated by one space in increasing order. Each integer is a vertex that the *Tarjan* and the *Trajan* algorithm may produce different results on. If there are no such vertices, output `Empty` in one line instead.

Example

standard input	standard output

2
9 12
1 3
2 3
1 2
3 6
6 9
9 3
4 6
5 6
4 5
7 9
8 9
7 8
2 1
1 2
	3 6 9
	Empty

Note

For the first sample test case:

- For $u = 3$ consider $v = 1$ and $adj = [[3, 2], [3, 1], [1, 2, 6, 9], [6, 5], [6, 4], [4, 5, 3, 9], [9, 8], [9, 7], [7, 8, 3, 6]]$.
- For $u = 6$ consider $v = 4$ and $adj = [[2, 3], [1, 3], [9, 6, 2, 1], [6, 5], [6, 4], [4, 5, 3, 9], [9, 8], [9, 7], [7, 8, 3, 6]]$.
- For $u = 9$ consider $v = 7$ and $adj = [[2, 3], [1, 3], [9, 6, 2, 1], [5, 6], [4, 6], [9, 3, 5, 4], [9, 8], [9, 7], [7, 8, 3, 6]]$.

Note that all adjs in the explanation are 1-indexed.

Problem K. Xiangqi

Chinese chess (Xiangqi) is a popular game enjoyed by people of all ages. This was even noted at the 49th ICPC World Finals.

Photo taken in the 49th ICPC World Finals

Today, having just learned the rules of Xiangqi, you challenge the grandmaster Nanani. The grandmaster does not want to overwhelm you, so you will play a simplified version of the game with the following rules:

- The chessboard has 9 columns and 10 rows. Only two pieces are placed: a Knight and a Rook. If a piece is at column x and row y, we denote its position by (x, y). Both the Rook and the Knight must stay within the chessboard and may not move outside it.
- The two players take turns moving their pieces. If, after a player's move, their piece moves to the exact position of the opponent's piece, it is said that they have "captured" the opponent's piece, and the game ends.
- The moving rules for the Knight are as follows:
  - It can move from (x_N, y_N) to $(x_N + 2, y_N \pm 1)$, if $(x_N + 1, y_N)$ is empty.
  - It can move from (x_N, y_N) to $(x_N - 2, y_N \pm 1)$, if $(x_N - 1, y_N)$ is empty.
  - It can move from (x_N, y_N) to $(x_N \pm 1, y_N + 2)$, if $(x_N, y_N + 1)$ is empty.
  - It can move from (x_N, y_N) to $(x_N \pm 1, y_N - 2)$, if $(x_N, y_N - 1)$ is empty.
- The Rook can move to any other position in the same row or column, but cannot jump over the other piece.
- Each player must move their piece on their turn. It can be proven that the current player always has a way to move their piece in any situation.

The grandmaster controls the Knight, and you control the Rook, with the grandmaster moving first. Does the Rook have a strategy such that, no matter how the Knight responds, the Rook can capture the Knight in finitely many moves without being captured by the Knight?

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 8010$) indicating the number of test cases. For each test case:

The first and only line contains four integers x_N, y_N, x_R and y_R ($1 \le x_N, x_R \le 9$, $1 \le y_N, y_R \le 10$), where (x_N, y_N) is the initial position of the Knight, and (x_R, y_R) is the initial position of the Rook. It is guaranteed that the initial positions of the two pieces are different.

Output

For each test case output one line. If such a strategy exists for the Rook, output `YES`; Otherwise, output `NO`.

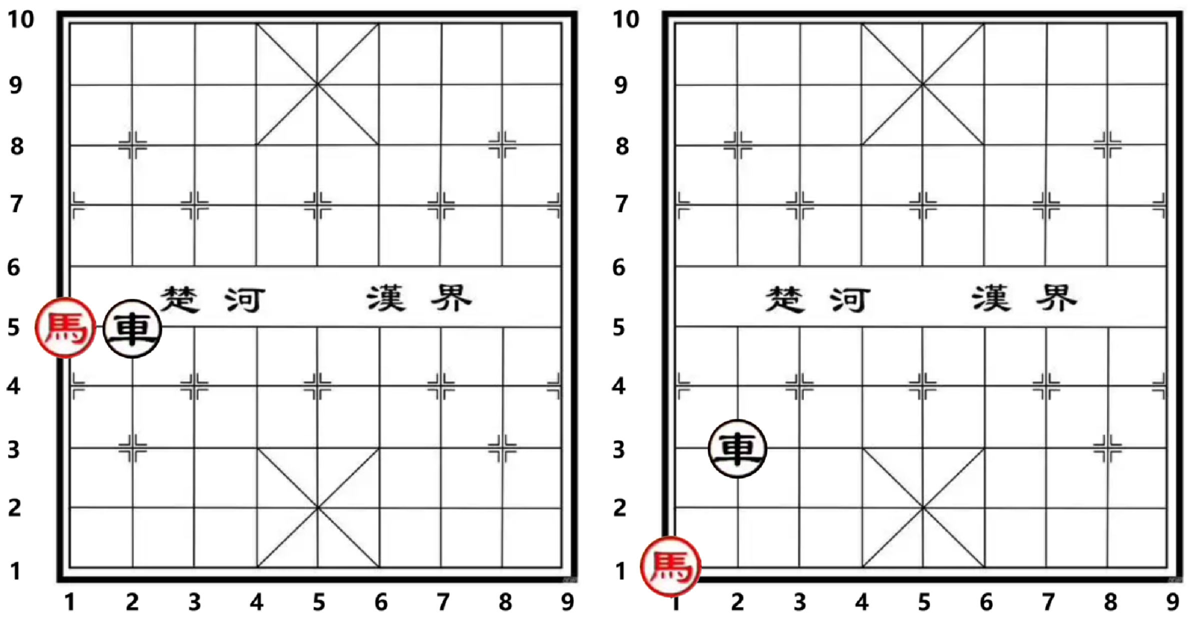
Example

standard input	standard output

2
1 5 2 5
1 1 2 3
	YES
	NO

Note

The sample test cases are shown as follows. Red pieces represent Knight, and black pieces represent Rook.



Problem L. Regional Champion

Arona wants to win the regional championship. So Plana generates the following problem for Arona based on the keyword “region”.

Given three integers n, m, and k, you need to draw n circles, m triangles, and k lines on a plane to divide it into as many regions as possible.

Of course, Arona’s drawing skills and mathematical abilities are very poor. So she comes to you for help.

Input

There are multiple test cases. The first line of the input contains an T ($1 \le T \le 5 \times 10^3$) indicating the number of test cases. For each test case:

The first and only line contains three integers n, m, and k ($0 \le n, m, k \le 100$, $n + m + k > 0$) indicating the number of circles, triangles, and lines.

It is guaranteed that the sum of $(\max(n, m, k))^3$ of all test cases does not exceed 10^7.

Output

For each test case:

First, output one line containing one integer, indicating the maximum number of regions that can be divided into.

Then output n lines. Each of these lines contains three integers x, y, and R ($-10^3 \le x, y \le 10^3$, $1 \le R \le 10^3$) separated by a space, representing the coordinates of the center of the circle, as well as the radius of the circle.

Next, output m lines. Each of these lines contains six integers x_1, y_1, x_2, y_2, x_3, and y_3 ($-10^3 \le x_1, y_1, x_2, y_2, x_3, y_3 \le 10^3$, $(x_1, y_1) \ne (x_2, y_2)$, $(x_1, y_1) \ne (x_3, y_3)$, $(x_2, y_2) \ne (x_3, y_3)$) separated by a space, representing the coordinates of the three vertices in the triangle.

At last, output k lines. Each of these lines contains four integers x_1, y_1, x_2, and y_2 ($-10^3 \le x_1, y_1, x_2, y_2 \le 10^3$, $(x_1, y_1) \ne (x_2, y_2)$) separated by a space, representing the coordinates of two points on the line.

It can be proven that, even with the limits specified above, we can always reach the optimal answer.

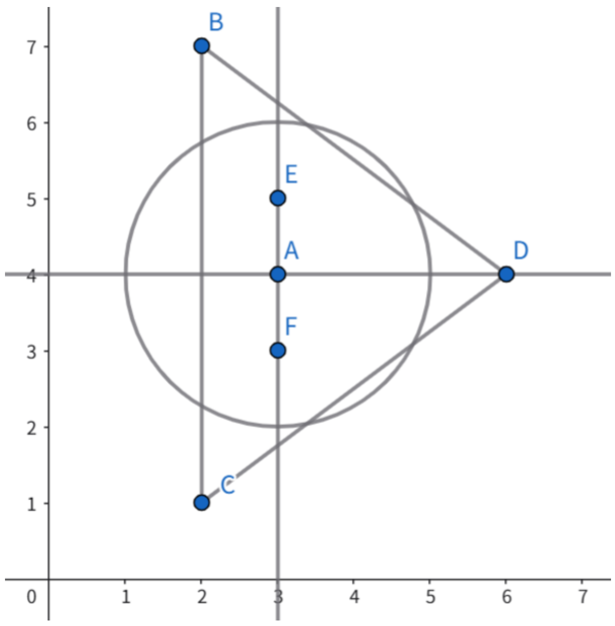
Example

standard input	standard output

3
1 0 1
0 1 1
1 1 2
	4
	0 0 1
	-2 0 2 0
	4
	0 0 0 2 2 0
	0 1 1 0
	18
	3 4 2
	2 7 2 1 6 4
	3 4 6 4
	3 3 3 5

Note

The third sample test case is shown below.



Problem M. Many Convex Polygons

Given the n vertices $P(0), P(1), \cdots, P(n-1)$ of a convex polygon in counter-clockwise order, there are $\binom{n}{k}$ ways to choose k vertices $P(i_0), P(i_1), \cdots, P(i_{k-1})$ $(0 \le i_0 < i_1 < \cdots < i_{k-1} < n)$. By connecting the selected vertices in counter-clockwise order (that is, for each $0 \le j < k$, connect vertices $P(i_j)$ and $P(i_{(j+1) \bmod k})$), a convex polygon with k vertices can be formed.

For each $3 \le k \le n$, consider all convex polygons with k vertices formed in the above manner. Calculate twice the sum of the areas of these convex polygons.

Input

There is only one test case in each test file.

The first line contains an integer n $(3 \le n \le 2 \times 10^5)$, indicating the number of vertices of the convex polygon.

For the following n lines, the i-th line contains two integers x_i and y_i $(-10^9 \le x_i, y_i \le 10^9)$, indicating the coordinates of the i-th vertex of the convex polygon. The vertices are given in counter-clockwise order, and no three vertices are collinear.

Output

Output $(n-2)$ lines, each containing one integer, where the i-th line is the answer to $k = i + 2$.

It can be proven that the answer is an integer. Since the answer may be large, output the answer modulo 998 244 353.

Example

standard input	standard output

6
0 0
1 0
2 1
2 2
1 2
0 1
	36
	54
	30
	6