ACM Collegiate Programming Contest 2018 (Hong Kong)

Judging Committee
6/23/2018

Venue:	City University of Hong Kong
Time:	2018-6-23 [Sat] 1330-1730
Number of Questions:	7
Number of Pages:	17
Input:	Standard Input
Output:	Standard Output
Memory Limit:	64 Megabytes

Problem A

Area Estimation

Time Limit: 8 seconds

Memory: 64 Megabytes

Given a convex shape drawn on a grid paper of unit squares, Kevin wants to estimate the area of the shape by counting the number of unit squares that are covered by the shape. See Figure 1 for an illustrative example. It is not so difficult when the shape is small, but can be challenging when it is large. Can you write a computer program to help Kevin counting correctly?

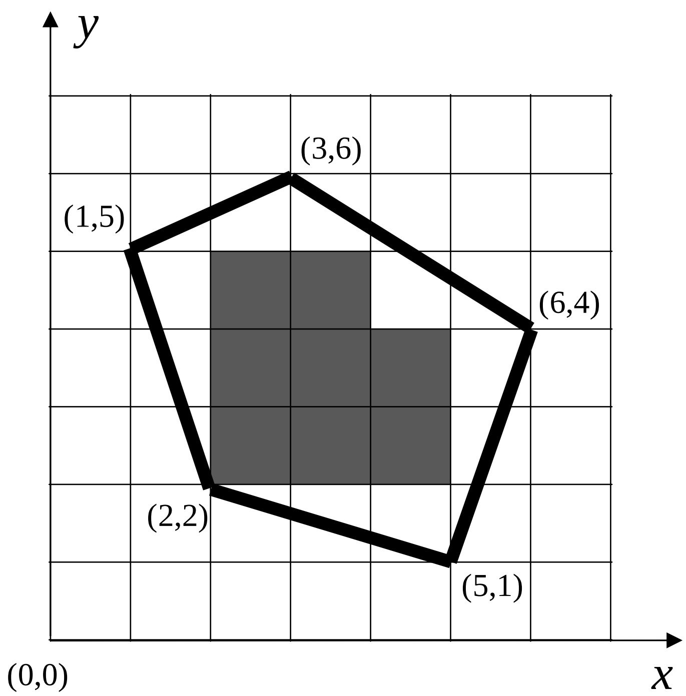

Figure 1. An illustrated example with a five-sided convex shape, which covers six unit squares.

Input

First line of the input is an integer T indicating the number of test cases. For each case, the first line contains an integer n, representing the number of vertices of a convex shape. The second line contains n pairs of integers, x_1, y_1, x_2, y_2, …, x_n, y_n, representing the coordinates for the n vertices of the convex shape in counterclockwise order, where $1 \le n \le 10000$, and the absolute values of the x and y coordinates are less than or equal to 1000000.

Output

For each case, output an integer representing the counting result, i.e., the number of unit squares covered by the shape.

Standard Input	Standard Output
1 5 2 2 5 1 6 4 3 6 1 5	8

Problem B

Box Puzzle

Time Limit: 10 seconds

Memory: 64 Megabytes

Owen likes to solve number puzzles. This time he is asked to fill n empty boxes located with positive integers. For each pair of boxes, the multiplication of their values are given. Can you please help Owen to determine whether or not the puzzle has a feasible solution? See Figure 2 for an illustrative example.

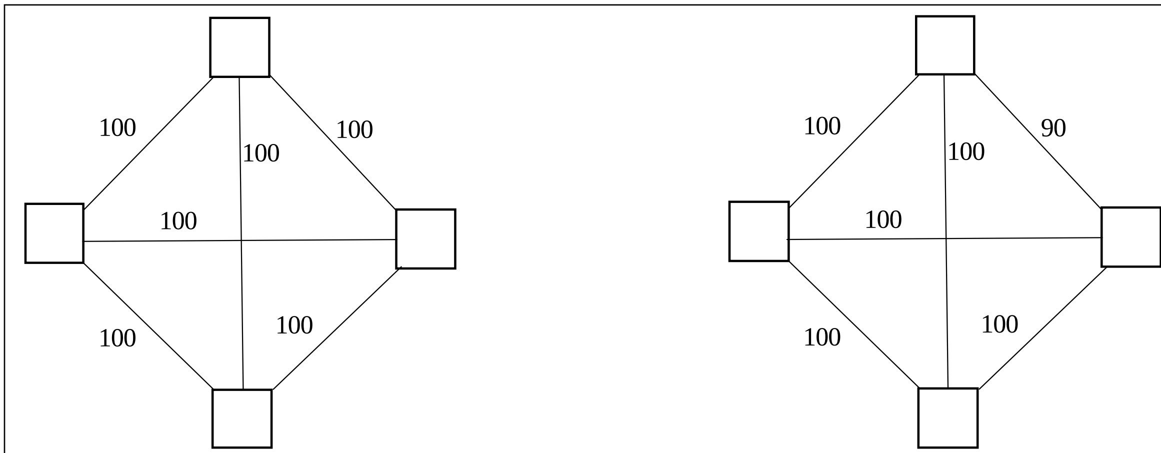

Figure 2. Two illustrated examples with the value of the multiplication located near to the line that links each pair of boxes, where the left example has a feasible solution, which fills each box with 10, and the right example has no feasible solution.

Input

First line of the input is an integer T indicating the number of test cases. For each case, its first line contains the integer n ($1 \leq n \leq 1000$). Each of the following n lines contains n-1 positive integers (which do not exceed 10^9) and a dummy value -1, where the j-th number of the i-th line representing the value of the multiplication of the values in box i and box j, and the j-th number of the j-th line is the dummy value -1.

Output

For each case, output “Yes” if the puzzle has a solution, and “No”, otherwise.

Standard Input	Standard Output
2 4 -1 100 100 100 100 -1 100 100 100 100 -1 100 100 100 100 -1 4 -1 100 100 90 100 -1 100 100 100 100 -1 100 100 100 100 -1	Yes No

Problem C

Chess with Only Pawns

Time Limit: 3 seconds

Memory: 64 Megabytes

Owen likes to play chess but with only pawns. The chess board, which has eight columns (denoted by A, B, C, …, and H) and eight rows (denoted by 1, 2, 3, …, and 8), has totally 64 spaces, and each space of the chess board can be represented by two characters indicating the column and the row, respectively. For example, A4 represents the space located at column A and row 4.

In the chess game with only pawns, there are two players with one player having white pawns, and the other player having black pawns. The side of the player with white pawns is row 1, and the side of the player with black pawns is row 8. Each player takes turns to move pawns forward towards the side of her opponent. The player with white pawns always moves first.

In each turn, one and only one pawn can move in the following two ways (see Figure 2.1):

1. The pawn can move one square forward to an unoccupied square immediately in front of it on the same column;
2. If an opponent's pawn occupies a square diagonally in front of it on an adjacent column, the pawn can move to that square, and can take the opponent's pawn in that square out of the chessboard. This is called "capture".

Except these, no other move is allowed.

The player with the first pawn that arrives at the side of her opponent, or with the opponent's pawns all captured wins the game. If either player cannot move any pawns, then no player wins the game. Encouraged by the success of Google's AlphaZero, Owen wants to build a computer program to play this game. To help on this, can you please write a program to determine that for some given initial positions of all the pawns, whether or not the player with white pawns, who will move next, has a winning strategy? See Figures 2.2 and 2.3 for two illustrative examples.

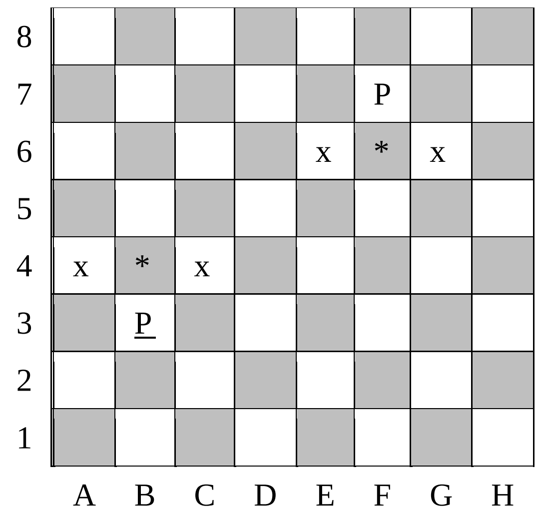

Figure 3.1. An illustrated example where P represents a white pawn, P represents a black pawn, * represents the squares immediately in front of them, and x represent squares square diagonally in front of them on adjacent columns.

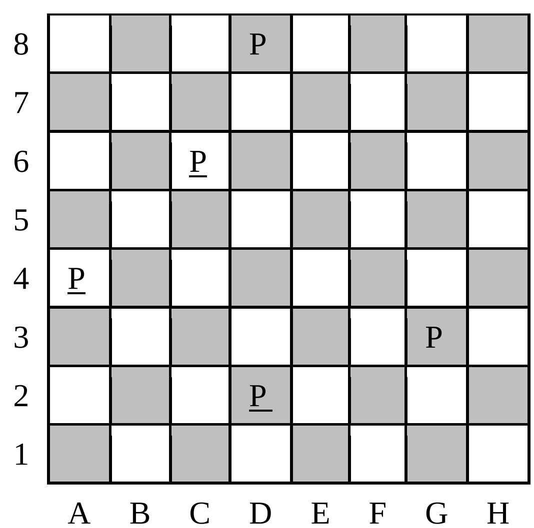

Figure 3.2. An illustrated example where P represents white pawns, P represents black pawns, and the player with white (Owen) has no winning strategy, since the white pawn at space C6 will be captured if it moves forward, so that the black pawn at space G3 will arrive at row 1 first.

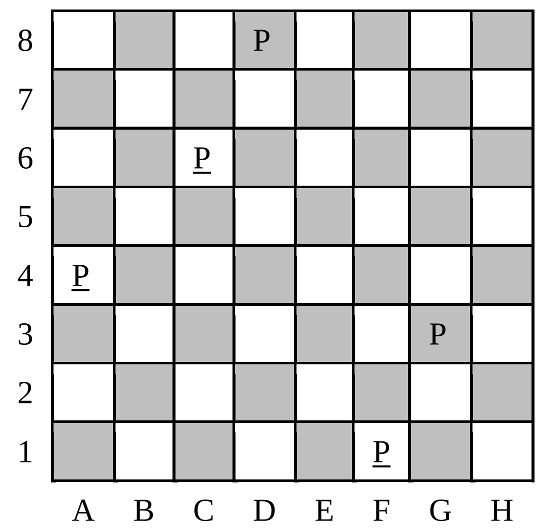

Figure 3.3. An illustrated example where $\underline{P}$ represent white pawns, P represent black pawns, and the player with white (Owen) has a winning strategy by moving the white pawn at A4 one space forward, so that both black pawns will be captured after they move forward.

Input

The first line of the input contains the number of test cases T. For each test case, its first line contains two integer N_0 *and* N_1, where $0 \leq N_0 \leq 4$ and $0 \leq N_1 \leq 4$. It is followed by two lines, with the first line including the initial positions of the N_0 white pawns, and with the second line including the initial positions of the N_1 black pawns. Each position is represented by a pair of column and row positions. Neither of the players has already won the game at the very beginning.

Output

For each test case, output one integer that equals 1 if Owen has a winning strategy, and equals 0 if Owen has not.

Example

Standard Input	Standard Output
5 2 1 D7 E4 E5 2 1 D6 E4 E5 3 2 A4 C6 D2 D8 G3 3 2 A4 C6 F1 D8 G3 1 1 A1 A8	1 0 0 1 0

Problem D

Destroying Squares

Time Limit: 1 seconds

Memory: 64 Megabytes

Kevin likes to play match-sticker puzzles. This time he is given *n* stickers of unit sizes, which are placed horizontally or vertically on sides of unit squares of a grid paper. No one sticker will overlap with another. Kevin is asked to remove a minimum number of match-stickers so that no unit squares on a grid are fully surrounded by match stickers. See Figure 3 for an illustrative example. Can you please write a program to tell Kevin the minimum number of match-stickers to be removed?

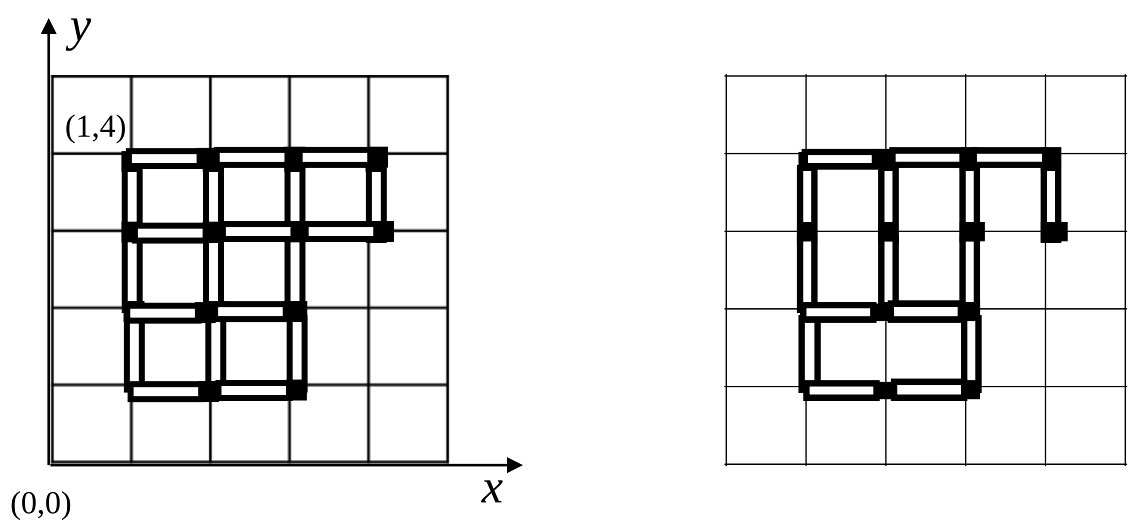

Figure 4. An illustrative example, where n=20 match-stickers are placed on a grid paper on the left, and at least four of them needs to be removed to avoid unit squares from being fully surrounded by math stickers, as shown on the right.

Input

The first line of the input contains the number of test cases T. For each test case, it starts with a line that contains an integers n where $1 \leq n \leq 10000$. It is then followed by n lines with each line i for $1 \leq i \leq n$ contains a triple (d_i, x_i, y_i), where d_i indicates the directions of the i-th match-sticker, which is horizontal if d_i=0, and is vertical if d_i=1, and where (x_i, y_i) indicates the location of match-sticker i, which is the coordinate of the left end of the i-th match-sticker i if d_i=0, and is the coordinate of the bottom end of the i-th match-sticker if d_i=1. The absolute values of the x and y coordinates are less than or equal to 1000000.

Output

For each test case, output the minimum number of match-stickers to be removed.

Example

Standard Input	Standard Output
1 20 0 1 4 0 2 4 0 3 4 1 1 3 1 2 3 1 3 3 1 4 3 0 1 3 0 2 3 0 3 3 1 1 2 1 2 2 1 3 2 0 1 2 0 2 2 1 1 1 1 2 1 1 3 1 0 1 1 0 2 1	4

Problem E

Egg Collection

Time Limit: 1 seconds

Memory: 64 Megabytes

Owen and Kevin play an egg collection game in an *xy*-plane, in which there are n eggs located at coordinates (x_i,y_i) for $i=1,2,\ldots,n$, where the x-axis points horizontally to the right, and the y-axis points vertically upwards. A robot is initially located at coordinate (0,0), facing upwards, and can either move either upward or rightward.

Owen needs to move the robot so as to collect at least m eggs, and then let his younger brother, Kevin, to choose and take away q eggs from the m eggs, where $1 \leq q \leq m$. For each egg i in $\{1,2,\ldots,n\}$, its value to Owen is denoted by v_i, and its value to Kevin is denoted by w_i. Both Owen and Kevin aim to maximize their own values. How shall Owen move the robot so that the total value of the collected eggs left to Owen is maximized? See Figure 3 for an illustrative example.

Input

The first line of the input contains the number of test cases T. Each test case has two lines. The first line contains the three integers n, m, q with $1 \leq q \leq m \leq n \leq 50$. The second line contains $4n$ integers, x_1, y_1, v_1, w_1, x_2, y_2, v_2, …, and x_n, y_n, v_n, w_n, whose absolute values do not exceed 10^9. It is guaranteed that each w_i is different.

Output

For each test case, output one line of an integer that equals -1 if the robot cannot collect at least m eggs, and equals the maximum possible total value of the collected eggs left to Owen, otherwise.

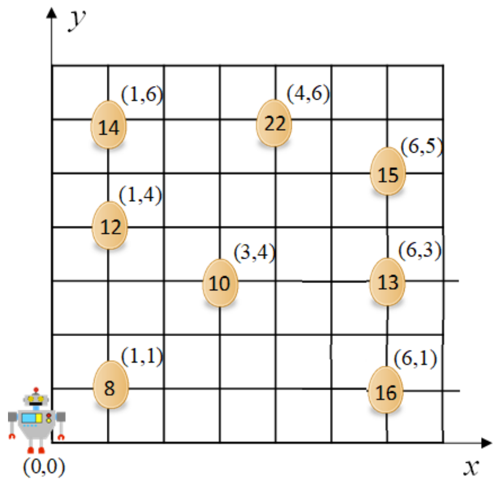

Figure 5. An illustrate example with n=8, m=3, and q=2, where the 8 eggs located at (1,1), (1,4), (1,6), (3,4), (6,3), (6,5), (6,1), and (4,6), with values to Owen and Kevin both equal to 8, 12, 14, 10, 13, 15, 16, and 22, and where Owen can move the robot from (0,0) to collect the 4 eggs at (1,1), (6,1), (6,3), and (6,5), so that if the 2 eggs at (6,5) and (6,1) are chosen and taken away by Kevin, the total value of the collected eggs left to Owen equals 8+13=21.

Example

Standard Input	Standard Output
2 8 3 2 1 1 8 8 1 4 12 12 1 6 14 14 3 4 10 10 6 3 13 13 6 5 15 15 6 1 16 16 4 6 22 22 8 5 2 1 1 8 8 1 4 12 12 1 6 14 14 3 4 10 10 6 3 13 13 6 5 15 15 6 1 16 16 4 6 22 22	21 -1

Problem F

Focus on Recovery

Time Limit: 1 seconds

Memory: 64 Megabytes

Consider N cities located on a circle, which are labeled clockwise by 0, 1, …, and N–1. For each city i in {0, 1, 2, …, N – 1}, there is a bidirectional link that connects city i and city (i + 1) mod N.

As an earthquake occurs, all the N cities have been destroyed. On each day i of the following Q days, a city C_i will send a team to recover a city. Given the first city C_1, each C_i with $2 \leq i \leq Q$ is determined recursively as follows:

$$C_i = [(C_{i-1} + 6655) \times 1551] \ mod \ N \ (2 \leq i \leq Q)$$

On each day i with $i = 1, 2, \ldots, Q$, city C_i will send its team to the first unrecovered city (denoted by U_i) in the clockwise direction from city C_i (including C_i). The team will recover city U_i, and then use Con_i to denote the number of cities that are connected to city U_i. Here, we define that two cities A and B are connected, if A and B are equal, or city A can be reached from city B by passing through links that connect only recovered cities.

After Q days, it is time for the cities to investigate the status of their recovery, which is measured by $\sum_{i=1}^{Q} Con_i$.

Input

In the first line, a number $\boldsymbol{T}$ ($\boldsymbol{T} \le \mathbf{10}$) is given, indicating the number of cases. For each case, three integers $\boldsymbol{N}, \boldsymbol{Q}, \boldsymbol{C_1}$ ($\mathbf{2} \le \boldsymbol{N} \le \mathbf{3000000}$; $\mathbf{1} \le \boldsymbol{Q} < \boldsymbol{N}$; $\mathbf{0} \le \boldsymbol{C_1} < \boldsymbol{N}$) are given, indicating the number of cities, the number of days for recovery, and the index of the city which send a team on the first day.

Output

For each case, output one line of an integer that represents the value of $\sum_{i=1}^{Q} Con_i$.

Standard Input	Standard Output
2 6 4 0 11 8 3	6 34

Problem G

Golden Sequence

Time Limit: 1 seconds

Memory: 64 Megabytes

A sequence of integers is a golden sequence if the absolute difference of any consecutive numbers in the sequence is a power of ten. For example,

- 1, 11, 21, 121, and 120 is a golden sequence;
- 1, 11, 21, 221, and 121 is not a golden sequence, because the absolute difference of 21 and 221 is 200, which is not a power of ten.

Kevin is given a sequence of numbers, but some of them are covered by ink accidentally. Can you please write a program to help him to decide whether or not the original number sequence is possible to be a golden sequence? See two illustrated examples below:

- Consider a given sequence 1, *, 21, *, and 120, where * indicates that the number is covered by ink. It can be verified that the original number sequence can be 1, 11, 21, 121, and 120, which is a golden sequence;
- Consider a given sequence 1, *, *, 21, *, and 121, where * indicates that the number is covered by ink. It can be verified that the original number sequence cannot be a golden sequence.

Input

The first line of the input contains the number of test cases T. For each test case, there is a single line of a sequence of positive numbers (which may contain leading 0s), and/or *, separated by a space, where * indicates that the number is covered by ink. Each line contains at most 10000 characters.

Output

For each test case, output “Yes” if the original number sequence can be a golden sequence, and output “No” otherwise.

Example

Standard Input	Standard Output
2 1 * 21 * 120 1 * * 21 * 121	Yes No