The 38th Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Korea Nationwide Internet Competition

Problem A
Battleship
Time Limit: 6 seconds

You are a member of the prestigious ICPC (Inter-Continental Protecting Corps) and fighting against the evil enemy. The battlefield is represented by an $n \times n$ grid. The coordinate of the leftmost and lowest point is $(1, 1)$ and that of the rightmost and highest point is (n, n). The enemy has a fleet of k battleships. Each battleship i is represented by a line segment whose length is greater than 0 and its endpoints are (x_i, y_i) and (x'_i, y'_i). Also, its weight is w_i. You can fire a laser cannon l times to destroy battleships. Each time you can fire it either horizontally or vertically. If you shoot it vertically, the laser is represented by a line linking $(a, 1)$ and (a, n) and all the battleships which meet the line (including the endpoints) are destroyed. If you shoot it horizontally, the laser is represented by a line linking $(1, a)$ and (n, a) and those which meet the line (including the endpoints) are destroyed. While you are a brave soldier, due to the inefficiency of bureaucracy, you are asked to report the heaviest battleship you just destroyed each time you shoot the laser cannon.

Consider the example in Figure 1. There are five battleships on a 5×5 grid. The weight of a battleship is written next to it. Assume that we first shoot the cannon vertically and the laser is a line linking $(4, 1)$ and $(4,5)$. Then two battleships are destroyed: one with weight 4 and the other with weight 5, which is heavier. Therefore we report 5. Next we shoot the cannon horizontally and the laser is a line linking $(1,4)$ and $(5,4)$. Again two battleships are destroyed: one with weight 1 and the other with weight 2, which is heavier. We report 2. Note that the battleship with weight 4 is already destroyed at the first shooting.

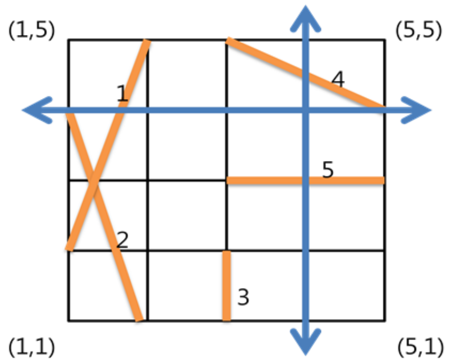

Figure 1. An example of battlefield with five battleships.

Given the locations of battleships and your shooting information, write a program which reports the heaviest battleship you destroyed each time you shoot the cannon.

Input
Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with integers n, k, and l, the size of the grid, the number of battleships, and the number of times you can shoot the laser cannon, respectively, where $1 \leq n \leq 1{,}000{,}000{,}000$ and $1 \leq k, l \leq 100{,}000$. Each of the following k lines contains five integers x, y, x', y' and w separated by a space which represent the endpoints of a battleship (x, y) and (x', y') $(1 \leq x, y, x', y' \leq n)$ and its weight $1 \leq w \leq 1{,}000{,}000$. Also each of the following l lines contains two integers a and b where $1 \leq a \leq n$. Either $b = 0$ or $b = 1$. If $b = 0$, you shoot the cannon horizontally and the laser is represented by a

line linking $(1, a)$ and (n, a). If $b = 1$, you shoot it vertically and the laser is represented by a line linking $(a, 1)$ and (a, n).

Output

Your program is to write to standard output. Print exactly l lines for each test case. Each line should contain an integer value, the weight of the heaviest battleship you destroyed by shooting the laser cannon. If you destroyed nothing, print 0.

The following shows sample input and output for two test cases.

Sample Input

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

Output for the Sample Input

```
5
2
3
0
```

Problem B
Cain Calendar
Time Limit: 1 second

It was recently revealed by the ICPC excavation team that the Inca Empire was established just after the Cain Empire which was a splendid civilization that flourished in South America. It is believed that the people in the Cain Empire used an interesting odd calendar. In their calendar, a year was represented by $< x{:}\, y >$, where x and y are natural numbers which are less than or equal to M and N, respectively. The first year, that is, the beginning of the world is represented by $< 1{:}\, 1 >$. The second year is represented by $< 2{:}\, 2 >$. Let $< x'{:}\, y' >$ be the following year of $< x{:}\, y >$. If $x < M$, $x' = x + 1$, otherwise $x' = 1$. Similarly, if $y < N$, $y' = y + 1$, otherwise $y' = 1$. The last year of their calendar is $< M{:}\, N >$. It is said that there was a prophecy which states the world ends in the year $< M{:}\, N >$.

For example, assume that $M = 10$ and $N = 12$. The first year is represented by $< 1{:}\, 1 >$. The year $< 1{:}\, 11 >$ represents the 11th year, $< 3{:}\, 1 >$ represents the 13th year, and $< 10{:}\, 12 >$ represents the 60th year which is the last year.

Given four integers M, N, x, and y, write a program that computes the number k such that $< x{:}\, y >$ represents the kth year, where $< M{:}\, N >$ is the last year of the world in the Cain Calendar.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case consists of a single line containing four integers M, N, x, and y ($1 \le M, N \le 40{,}000$, $1 \le x \le M$, $1 \le y \le N$), where $< M{:}\, N >$ is the last year of the world in the Cain Calendar.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain an integer k, where the kth year is represented by $< x{:}\, y >$ for x and y given in the input. If there doesn't exist a year represented by $< x{:}\, y >$, print `-1`.

The following shows sample input and output for three test cases.

Sample Input

```
3
10 12 3 9
10 12 7 2
13 11 5 6
```

Output for the Sample Input

```
33
-1
83
```

The 38th Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Korea Nationwide Internet Competition

Problem B
카잉 달력
Time Limit: 1 second

최근에 ICPC 탐사대는 남아메리카의 잉카 제국이 놀라운 문명을 지닌 카잉 제국을 토대로 하여 세워졌다는 사실을 발견했다. 카잉 제국의 백성들은 특이한 달력을 사용한 것으로 알려져 있다. 그들은 M 과 N 보다 작거나 같은 두 개의 자연수 x, y를 가지고 각 년도를 $<x:y>$와 같은 형식으로 표현하였다. 그들은 이 세상의 시초에 해당하는 첫 번째 해를 $<1:1>$로 표현하고, 두 번째 해를 $<2:2>$로 표현하였다. $<x:y>$의 다음 해를 표현한 것을 $<x':y'>$이라고 하자. 만일 $x < M$ 이면 $x' = x + 1$이고, 그렇지 않으면 $x' = 1$이다. 같은 방식으로 만일 $y < N$이면 $y' = y + 1$이고, 그렇지 않으면 $y' = 1$이다. $<M:N>$은 그들 달력의 마지막 해로서, 이 해에 세상의 종말이 도래한다는 예언이 전해 온다.

예를 들어, $M = 10$ 이고 $N = 12$라고 하자. 첫 번째 해는 $<1:1>$로 표현되고, 11 번째 해는 $<1:11>$로 표현된다. $<3:1>$은 13 번째 해를 나타내고, $<10:12>$는 마지막인 60 번째 해를 나타낸다.

네 개의 정수 M, N, x 와 y가 주어질 때, $<M:N>$이 카잉 달력의 마지막 해라고 하면 $<x:y>$는 몇 번째 해를 나타내는 지를 구하는 프로그램을 작성하라.

입력(Input)
입력 데이터는 표준입력을 사용한다. 입력은 T개의 테스트 데이터로 구성된다. 입력의 첫 번째 줄에는 입력 데이터의 수를 나타내는 정수 T가 주어진다. 각 테스트 데이터는 한 줄로 구성된다. 각 줄에는 네 개의 정수 M, N, x 와 y가 주어진다($1 \le M, N \le 40{,}000$, $1 \le x \le M$, $1 \le y \le N$), 여기서 $<M:N>$은 카잉 달력의 마지막 해를 나타낸다.

출력(Output)
출력은 표준출력을 사용한다. 각 테스트 데이터에 대해, 정수 $\boldsymbol{k}$를 한 줄에 출력한다. 여기서 $\boldsymbol{k}$는 $<\boldsymbol{x}:\boldsymbol{y}>$가 $\boldsymbol{k}$번째 해를 나타내는 것을 의미한다. 만일 $<x:y>$에 의해 표현되는 해가 없다면, 즉, $<x:y>$가 유효하지 않은 표현이면, -1을 출력하라.

다음은 세 개의 테스트 데이터에 대한 입력과 출력의 예이다.

입력 예제(Sample Input)	**출력 예제(Output for the Sample Input)**

`3`
`10 12 3 9`
`10 12 7 2`
`13 11 5 6`
	`33`
	`-1`
	`83`

Problem C
Casting
Time Limit: 10 seconds

Casting is a manufacturing process in which liquid is poured into a cast that has a cavity with the shape of the object to be manufactured. The liquid then hardens, after which the cast is removed. To ensure that the given object can be mass produced by re-using the same case parts, the cast parts must be removed from the object without destroying either the cast parts or the object.

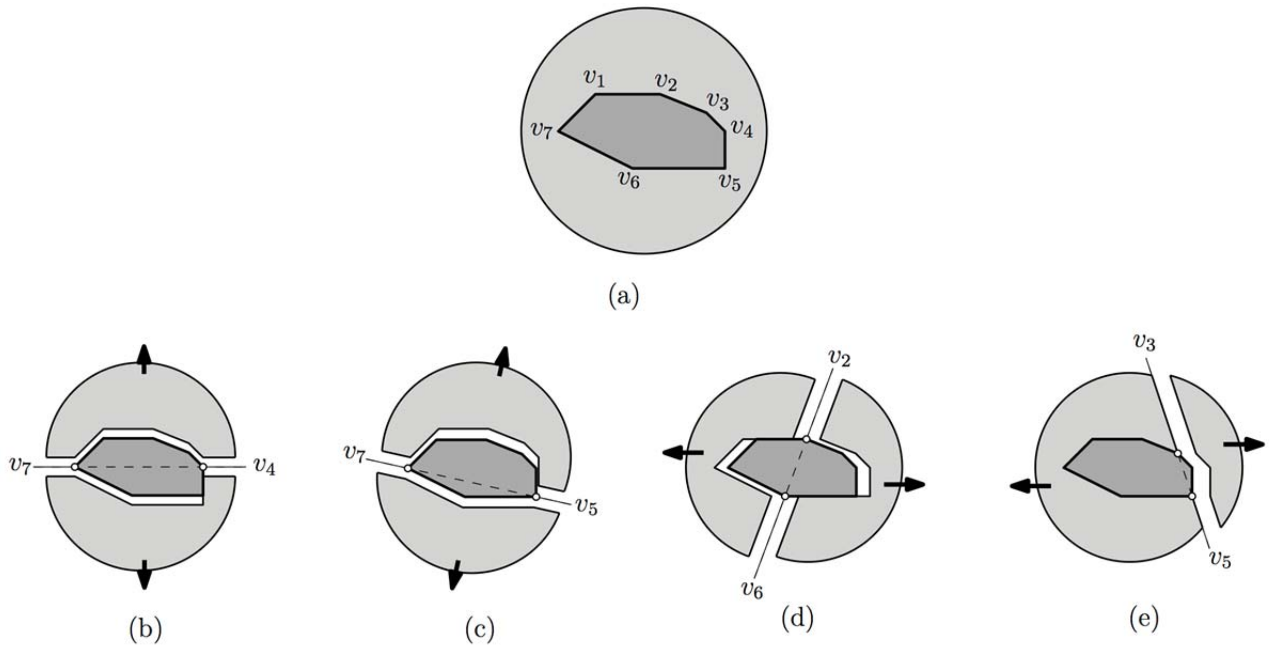

Figure 1.

Figure 1(a) shows an object (dark gray) with its cast (light gray). Our goal is to divide the cast into two parts along a straight line through two vertices of the object such that the cast parts can be removed from the object by translation without destroying either the cast parts or the object. In Figure 1(b), the cast is divided into two parts by the straight line through v_4 and v_7 such that the upper part is removed vertically upward and the lower part is removed vertically downward from the object without destroying either the cast parts or the object. Figure 1(c) and Figure 2(d) show such divisions by the straight line through v_5 and v_7, and by the straight line through v_2 and v_6, respectively. However, not every pair of vertices defines such a division. In Figure 1(e), the left part defined by the straight line through v_3 and v_5 cannot be removed from the object without destroying either the cast parts or the object.

Given a convex polygon P with n vertices in the plane, your program is to find all pairs (v_i, v_j) of vertices of P such that both cast parts of P divided by the straight line through (v_i, v_j) can be removed by translation without destroying either the cast parts or the object.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with integer n, the number of vertices of a convex polygon P, where $3 \le n \le 100{,}000$. The next line contains a sequence of $2n$ integers, $x_1\ y_1\ x_2\ y_2 \ldots x_n\ y_n$, where x_i and y_i are the x-coordinate and y-coordinate of vertex v_i of P, respectively. The coordinates are all integers with $-1{,}000{,}000{,}000 \le x_i \le 1{,}000{,}000{,}000$ and $-1{,}000{,}000{,}000 \le y_i \le 1{,}000{,}000{,}000$. Vertices $v_1, v_2, \ldots, v_n$ of P are given in clockwise order along the boundary of P.

Output

Your program is to write to standard output. For each test case, print a line containing the number of pairs (v_i, v_j) of vertices with $i < j$ such that both cast parts of P divided by the straight line through (v_i, v_j) can be removed by translation without destroying either the cast parts or the object.

The following shows sample input and output for three test cases.

Sample Input

```
3
3
0 0 3 3 6 0
4
0 0 0 2 5 2 5 0
5
0 0 3 3 3 2 3 1 3 0
```

Output for the Sample Input

```
3
6
10
```

The 38th Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Korea Nationwide Internet Competition

Problem D
Dual Priority Queue
Time Limit: 3 seconds

A dual priority queue is a data structure in which we can insert and delete data like a typical queue. Its difference from a typical queue is that when an item is to be deleted from the queue either an item with highest priority or with lowest priority in it is deleted according to the operation for deletion. There are two operations for the dual priority queue, one for inserting an item into it, the other for deleting an item from it. Deletion operation itself has two parameters: one for deleting an item with highest priority from it, and the other for deleting an item with lowest priority.

Suppose there is a dual priority queue Q which stores integers only. The value of each integer in Q is considered the priority of itself.

Given a list of operations, write a program that processes the operations for Q and shows the minimum and maximum numbers in Q after finishing the operations.

Input
Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with a line containing an integer k ($k \leq 1{,}000{,}000$), the number of operations for Q. In the next k lines, each line contains either '`D`' or '`I`' followed by an integer n. Operation '`I` n' means inserting integer n into Q. Note that the same integers can be inserted into Q. Operation '`D 1`' means deleting the maximum number from Q. While Operation '`D` -1' is used to delete the minimum number from Q. If there are two or more maxima (minima) in Q when the operation is '`D 1`'('`D` -1'), only one of them is deleted.

If Q is empty when the operation is '`D`' you can ignore it. Each input to be inserted into Q is a 32-bit integer.

Output
Your program is to write to standard output. For each test case, print the maximum and the minimum numbers among those which remain in Q. The two numbers should be printed in a line separated by a blank. If Q is empty at the end, print '`EMPTY`'.

The following shows sample input and output for two test cases.

Sample Input
```
2
7
I 16
I -5643
D -1
D 1
D 1
I 123
```

Output for the Sample Input
```
EMPTY
333 -45
```

D -1 9 I -45 I 653 D 1 I -642 I 45 I 97 D 1 D -1 I 333	

Problem D
이중 우선순위 큐
Time Limit: 3 seconds

이중 우선순위 큐(dual priority queue)는 전형적인 우선순위 큐처럼 데이터를 삽입, 삭제할 수 있는 자료 구조이다. 전형적인 큐와의 차이점은 데이터를 삭제할 때 연산(operation) 명령에 따라 우선순위가 가장 높은 데이터 또는 가장 낮은 데이터 중 하나를 삭제하는 점이다. 이중 우선순위 큐를 위해선 두 가지 연산이 사용되는데, 하나는 데이터를 삽입하는 연산이고 다른 하나는 데이터를 삭제하는 연산이다. 데이터를 삭제하는 연산은 또 두 가지로 구분되는데 하나는 우선순위가 가장 높은 것을 삭제하기 위한 것이고 다른 하나는 우선순위가 가장 낮은 것을 삭제하기 위한 것이다.

정수만 저장하는 이중 우선순위 큐 Q가 있다고 가정하자. Q에 저장된 각 정수의 값 자체를 우선순위라고 간주하자.

Q에 적용될 일련의 연산이 주어질 때 이를 처리한 후 최종적으로 Q에 저장된 데이터 중 최대값과 최솟값을 출력하는 프로그램을 작성하라.

입력(Input)
입력 데이터는 표준입력을 사용한다. 입력은 $\boldsymbol{T}$개의 테스트 데이터로 구성된다. 입력의 첫 번째 줄에는 입력 데이터의 수를 나타내는 정수 $\boldsymbol{T}$가 주어진다. 각 테스트 데이터의 첫째 줄에는 Q에 적용할 연산의 개수를 나타내는 정수 k ($k \leq 1{,}000{,}000$)가 주어진다. 이어지는 k 줄 각각엔 연산을 나타내는 문자(‘D’ 또는 ‘I’)와 정수 n이 주어진다. ‘I n’은 정수 n을 Q에 삽입하는 연산을 의미한다. 동일한 정수가 삽입될 수 있음을 참고하기 바란다. ‘D 1’는 Q에서 최대값을 삭제하는 연산을 의미하며, ‘D -1’는 Q에서 최솟값을 삭제하는 연산을 의미한다. 최대값(최솟값)을 삭제하는 연산에서 최대값(최솟값)이 둘 이상인 경우, 하나만 삭제됨을 유념하기 바란다.

만약 Q가 비어있는데 적용할 연산이 ‘D’라면 이 연산은 무시해도 좋다. Q에 저장될 모든 정수는 32-비트 정수이다.

출력(Output)

출력은 표준출력을 사용한다. 각 테스트 데이터에 대해, 모든 연산을 처리한 후 Q에 남아 있는 값 중 최대값과 최솟값을 출력하라. 두 값은 한 줄에 출력하되 하나의 공백으로 구분하라. 만약 Q가 비어있다면 ‘EMPTY’를 출력하라

다음은 두 개의 테스트 데이터에 대한 입력과 출력의 예이다.

입력 예제(Sample Input)

```
2
7
I 16
I -5643
D -1
D 1
D 1
I 123
D -1
9
I -45
I 653
D 1
I -642
I 45
I 97
D 1
D -1
I 333
```

출력 예제(Output for the Sample Input)

```
EMPTY
333 -45
```

Problem E
Falling Ants
Time Limit: 1 second

N ants are on a stick where some facing right and some facing left. You can assume all ants are so tiny compared to the distance between them, that they can be considered moving points without volume. From a start signal, all ants begin to march in whichever direction they are currently facing. All ants march in a constant speed such as $1mm$ per second. If two different ants collide on a point, then they bounce and reverse their previous direction. Bouncing and reversing movement does not take any time. When an ant is reaching the end of stick, that ant falls off from the stick and down to the ground. We assume the stick is floating over a floor.

Initially all ants are placed in distinct positions, that means no two ants are placed at a same point of the stick. We represent each ant using a signed integer, called an ant ID. The sign of the ant ID denotes the facing direction in the intial state, where `'-'('+')` means facing the left (right). The absolute value of the ant ID is one of N integers $1, 2, \cdots, N$. So the absolute values of the ant ID are all distinct. In Figure 1, you can see that there are 6 ants with the signed ID $\{+4, +5, -1, -3, -2, +6\}$ whose corresponding positions are $\{5,8,19,22,24,25\}$ on a long stick with length $L = 30$. The arrow assigned for each ant shows the facing direction in the initial state. The position of the left end is 0, and the position of the right end is 30. It is easy to see that the ant of ID +6 will arrive at the right end of the stick at time $t = 5$, and then it falls off the stick at $t = 6$.

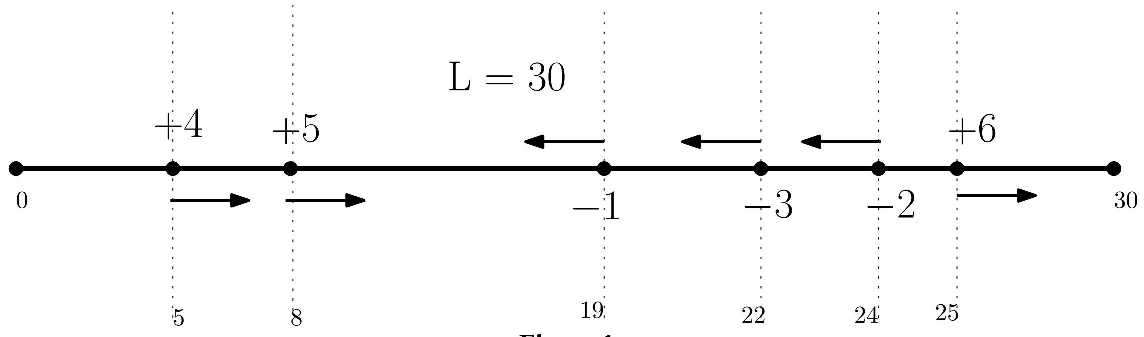

Figure 1.

You are given information of ants before marching; ant ID and the corresponding position on a stick. If two ants are falling simultaneously from both sides (left and right), then we will break the tie of falling order such that the ant with smaller ID number falls off slightly earlier than the other. Let us give one example for this case. In Figure 2, if two ants with ID = $\{-1, +2\}$ will reach each end simultaneously, the ant of ID = -1 will fall off earlier than the ant of ID = $+2$ since $-1 < +2$. So the falling sequence of four ants in Figure 2 is $\langle -1, 2, 4, 3 \rangle$, i.e., the ant of ID = -1 falls off the first and the ant of ID = 3 falls off the last.

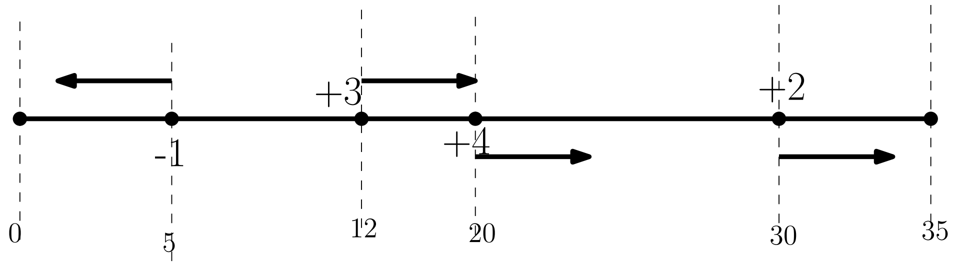

Figure 2.

Given a positive integer $1 \leq k \leq n$, you should find the k-th ant in the falling sequence, i.e., the k-th falling ant.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with three integer numbers N, L and k, where N is the total number of the ants, L is the length of the stick, and k is the falling order we are concerned with $k \leq N$. Each line in the following has two integer numbers, p_i and a_i , where p_i is the initial position of the ant a_i. Note that p_i is increasing such as $p_i < p_{i+1}$. Note that $1 \leq p_i \leq L-1$, $3 \leq N \leq 100{,}000$, $10 \leq L \leq 5{,}000{,}000$ and $1 \leq k \leq N$.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain the ID number of the k-th falling ant from the stick among all N ants. You should not write '+' symbol if the ant ID is positive.

The following shows sample input and output for two test cases which are the description of Figure 1 and 2.

Sample Input	**Output for the Sample Input**
2	-2
6 30 3	2
5 4	
8 5	
19 -1	
22 -3	
24 -2	
25 6	
4 35 2	
5 -1	
12 3	
20 4	
30 2	

The 38th Annual ACM

International Collegiate Programming Contest

Asia Regional – Daejeon

Korea Nationwide Internet Competition

Problem F

KCPC

Time Limit: 1 second

You are participating in the prestigious programming competition called KCPC (Korean Collegiate Programming Contest). You are supposed to solve k problems and for each problem you get between 0 and 100 points each time you submit your solution to the server. The submission is done electronically and the server maintains a log in order of submission time. The log includes your team ID, the problem number, and your score. For any problem you can submit your solution several times and you get the greatest points among your submissions. (For a problem, you get 0 points if you make no submission for the problem.)

Your final score is simply the sum of the points you earned and your ranking is (the number of teams with higher score than yours) + 1.

However, there may be two or more teams with the same score and the tie is broken by applying the ground rules in the following order.

Rule 1: If the final scores are the same, the team with the smaller number of submissions in total wins.
Rule 2: If both the final scores and the numbers of submission are the same, the team whose last submission is earlier wins.

Assume that no two solutions are submitted at the same time and every team makes at least one submission.

Given the log of the server, write a program to compute your ranking.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with four integers, the number of teams n, the number of problems k, your team ID t, and the number of log entries m, where $3 \leq n, k \leq 100$, $1 \leq t \leq n$, and $3 \leq m \leq 10{,}000$. In the following m lines, the information on submission is given in order of submission time. Each line contains three integers, team ID i, problem number j, and score s, where $1 \leq i \leq n$, $1 \leq j \leq k$, and $0 \leq s \leq 100$.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain your ranking.

The following shows sample input and output for two test cases.

Sample Input	Output for the Sample Input

2
3 4 3 5
1 1 30
	1
	2

```
2 3 30
1 2 40
1 2 20
3 1 70
4 4 1 10
1 1 50
2 1 20
1 1 80
3 1 0
1 2 20
2 2 10
4 3 0
2 1 0
2 2 100
1 4 20
```

The 38th Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Korea Nationwide Internet Competition

Problem F
KCPC
시간 제한: 1 초

당신은 유명 프로그래밍 대회인 KCPC(Korean Collegiate Programming Contest)에 참가하고 있다. 이 대회에서 총 k개의 문제를 풀게 되는데, 어떤 문제에 대한 풀이를 서버에 제출하면 그 문제에 대해 0점에서 100점 사이의 점수를 얻는다. 풀이를 제출한 팀의 ID, 문제 번호, 점수가 서버의 로그에 제출되는 시간 순서대로 저장된다. 한 문제에 대한 풀이를 여러 번 제출할 수 있는데, 그 중 최고 점수가 그 문제에 대한 최종 점수가 된다. (만약 어떤 문제에 대해 풀이를 한번도 제출하지 않았으면 그 문제에 대한 최종 점수는 0점이다.)

당신 팀의 최종 점수는 각 문제에 대해 받은 점수의 총합이고, 당신의 순위는 (당신 팀보다 높은 점수를 받은 팀의 수)+1 이다.

점수가 동일한 팀이 여럿 있을 수 있는데, 그 경우에는 다음 규칙에 의해서 순위가 정해진다.

규칙 1. 최종 점수가 같은 경우, 풀이를 제출한 횟수가 적은 팀의 순위가 높다.
규칙 2. 최종 점수도 같고 제출 횟수도 같은 경우, 마지막 제출 시간이 더 빠른 팀의 순위가 높다.

동시에 제출되는 풀이는 없고, 모든 팀이 적어도 한 번은 풀이를 제출한다고 가정하라.

서버의 로그가 주어졌을 때, 당신 팀의 순위를 계산하는 프로그램을 작성하시오.

입력(Input)
입력 데이터는 표준 입력을 사용한다. 입력은 T개의 테스트 데이터로 구성된다. 입력의 첫 번째 줄에는 테스트 데이터의 수를 나타내는 정수 T가 주어진다. 각 테스트 데이터의 첫 번째 줄에는 팀의 개수 n, 문제의 개수 k, 당신 팀의 ID t, 로그 엔트리의 개수 m을 나타내는 4 개의 정수가 주어진다. 여기서, $3 \leq n, k \leq 100$, $1 \leq t \leq n$, $3 \leq m \leq 10{,}000$이다. 그 다음 m개의 줄에는 각 풀이에 대한 정보가 제출되는 순서대로 주어진다. 각 줄에는 팀 ID i, 문제 번호 j, 획득한 점수 s를 나타내는 세 개의 정수가 주어진다. 여기서 $1 \leq i \leq n$, $1 \leq j \leq k$, $0 \leq s \leq 100$이다.

출력(Output)

출력은 표준출력을 사용한다. 주어진 각 테스트 데이터에 대해 당신 팀의 순위를 한 줄에 출력하여야 한다.

다음은 두 개의 테스트 데이터에 대한 입력과 출력의 예이다.

입력 예제(Sample Input)	출력 예제(Output for the Sample Input)

2
3 4 3 5
1 1 30
2 3 30
1 2 40
1 2 20
3 1 70
4 4 1 10
1 1 50
2 1 20
1 1 80
3 1 0
1 2 20
2 2 10
4 3 0
2 1 0
2 2 100
1 4 20
	1
	2

Problem G
Moore Machine
Time Limit: 5 seconds

Moore Machine is a finite state machine of which the output is determined by the states of them. The Moore Machine is named after Edward F. Moore, an American mathematician and computer scientist. The state transtion of a Moore Machine is directed by the given input. For example, if the input is given as “aabba,” the output of the following Moore Machine is “PRETTY.”

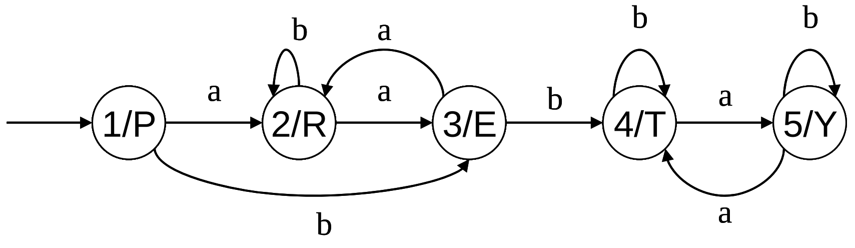

Figure 1. An example of a Moore Machine

In Figure 1, a circle denotes each state and the label on the arrow denotes the input symbol. One of the states is designated as the start state, which is represented by an arrow with no origin pointing to the state. In this case, the start state is State 1. Each state N with output symbol S is depicted in label N/S.

Generally, a Moore Machine can have cycles. However, we can think of a certain kind of Moore Machine which has no cycle at all. Let’s call this kind of Moore Machine a series-parallel Moore Machine.

Unfortunately, one of the output symbols of a series-parallel Moore Machine is erased. The problem is to find the erased output symbol using the given output of the machine. Note that it is not always possible to find the erased symbol. For example, assume the following series-parallel Moore Machine is given.

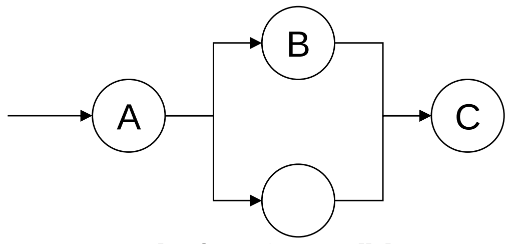

Figure 2. An example of a series-parallel Moore Machine

In Figure 2, only the output symbols are shown as labels on states for simplicity. The white circle without any label denotes the state whose output symbol is erased. If the given output of the machine is “ADC,” then we can determine the erased symbol is D. However, if the output is “ABC,” we cannot uniquely determine the erased symbol. Or, if the output is given as “ABD,” it is not a valid output of this machine. Your program should also determine these *impossible* cases.

Since the Moore Machine is reduced to a series-parallel graph, the representation of the machine itself can be defined in a straight-forward way. A Moore Machine consists of a single state, whose output is S, is denoted by S itself. A Moore Machine consists of a single state, whose output symbol is erased, is denoted by the underscore symbol ‘_’. The Moore Machine consists of a series of sub-machines, whose descriptions are M_1, M_2, …, and M_k respectively, is denoted by $M_1 M_2 \dots M_k$. The Moore Machine consists of a parallel connection sub-machines, whose descriptions are M_1, M_2, …, and M_k respectively, is denoted by $M_1|M_2|\dots|M_k$. If a Moore Machine is used as a part of another bigger Moore Machine, the parentheses can be used for enclosing the part. Using this representation, the above Moore Machine can be given as A(B|_)C.

Write a program determining the erased symbol for a given series-parallel Moore Machine and an output produced by the machine. If the erased symbol cannot be uniquely determined, print the underscore symbol ‘_’ instead. If the output cannot be produced from the given machine, print the exclamation mark ‘!’ instead.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. From the second line, each test case is given. A test case consists of two lines. The string in the first line of a test case contains the representation of the series-parallel Moore Machine. The second line contains an output of the machine. Only capital English alphabets are used for the output symbols. It is assumed that the outermost connection of the Moore Machine is serial. It is also assumed that there is only one erased symbol in the input machine. The lengths of the input lines of test cases are less than or equal to 100.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain the erased output symbol. If it cannot be determined, the underscore symbol should be printed. If the output cannot be produced from the machine, the exclamation mark symbol should be printed.

The following shows sample input and output for three test cases.

Sample Input	Output for the Sample Input
3 A(B|_)C ADC A(B|D|_)C ADC A(B|CD(E|_)G)E BOY	D _ !

Problem H
Networks with Unidirectional Links
Time Limit: 1 second

There is a multicomputer system that consists of a number of nodes, each with its own memory. The nodes in the system are interconnected via unidirectional communication links. The interconnection network of the system, representing the way in which the nodes are connected together, is modeled as a directed graph, where vertices and directed edges respectively represent nodes and unidirectional links of the network. Figure 1 shows an example of interconnection networks with fourteen nodes interconnected via nineteen unidirectional links.

Linear arrays and rings are two of the important computational structures in interconnection networks. In a linear array, each node except for the extreme ends has two neighboring nodes, one to its left and one to its right, where the two nodes at the ends are connected to their single neighbor. If the nodes at either end are connected, we refer to it as a ring. More specifically, the nodes of a linear array with k nodes can be numbered from 0 to $k-1$ so that there exists a unidirectional link from node i to node i+1 for every $0 \le i < k-1$. The linear array becomes a ring if we add a unidirectional link from the node $k-1$ to the node 0.

To support parallel applications that require one of the two aforementioned computational structures, we needs to decompose the system into several subsystems each of whose interconnection networks is a ring or a linear array. The subsystems should be pairwise node-disjoint, i.e., no two subsystems share a common node. According to a recent report, a ring composed of k nodes is worth k dollars while a linear array of k nodes is worth $k-1$ dollars. This motivates the study on how to decompose the system into subsystems in order to maximize profits. You are to write a program for decomposing the interconnection network of our multicomputer system into rings and/or linear arrays whose total value is the maximum possible. Note that a linear array may have only one node and in that case, its value is zero dollar.

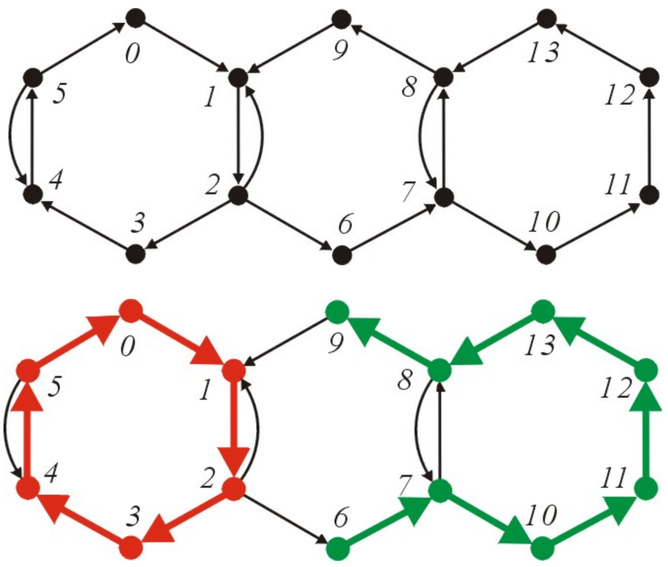

Figure 1. An interconnection network and its decomposition into a ring of six nodes (red) and a linear array of eight nodes (green). The total value of this decomposition is thirteen dollars, which is the maximum possible.

Input
Your program is to read from standard input. The input consists of T test cases, where the positive integer T is given in the first line of the input, followed by the description of each test case. The first line of a test case contains two positive integers n and m, respectively indicating the numbers of nodes and unidirectional links in an interconnection network, in which we assume $n \leq 1{,}000$ and $m \leq 50{,}000$. The nodes are indexed 0 to $n-1$. In the following m lines, each line contains two integers u and v, which represent a unidirectional link from node u to node v. The two integers given in a single line are always separated by a space.

Output
Your program is to write to standard output. Print exactly one line for each test case. The line should contain an integer, representing the maximum total value in dollars that can be achieved by decomposing the interconnection network into rings and/or linear arrays.

The following shows sample input and output for three test cases.

Sample Input

```
3
4 3
3 2
1 0
2 3
6 6
0 1
1 2
2 3
3 1
3 4
4 5
14 19
0 1
1 2
2 3
3 4
4 5
5 0
5 4
2 1
2 6
6 7
7 8
8 9
9 1
8 7
7 10
10 11
11 12
12 13
13 8
```

Output for the Sample Input

```
3
5
13
```

The 38th Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Korea Nationwide Internet Competition

Problem I
Pickup Game
Time Limit: 10 seconds

You are playing a computer game called the "Pickup." The rules are as follows:

(1) There are a number of horizontal and vertical line segments. (See the figure below.)
(2) You can "Pickup" a pair of line segments by clicking on the crossing made by the pair. (The pair disappears from screen when you do this.)
(3) Each line segment is given a weight.
(4) When you pick up a pair with weights a and b, you are given a score of $a \times b$.

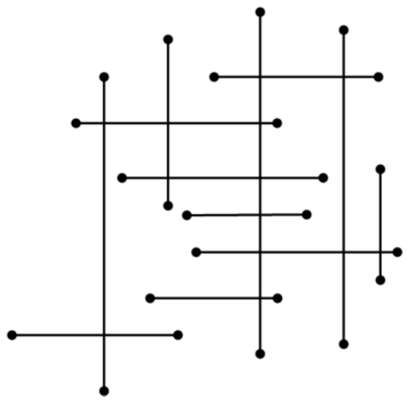

Figure 1. An Example Screen

The first goal is to pick up the most line segments. The second goal is to maximize the total score. That is, if there are more than one way of picking up the most line segments, then you have to find the one with the largest possible score.

Write a program, given the specification of an instance of the game, which computes the maximum possible number of line segments that can be picked up and the maximum possible score.

Input
Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with integers n and m, the number of horizontal and vertical line segments, respectively, where $1 \leq n, m \leq 200$. Each of the following n lines contains five integers x, y, x', y', w, representing the two endpoints (note, $y = y'$) and the weight of each horizontal line segment. Also each of the following m lines contains an analogous information for each vertical line segment. All coordinate values are positive and are less than or equal to $100{,}000$. The weights are also positive and are less than or equal to 20. Any pair of line segments may share at most one point which is never an endpoint of the line segments.

Output

Your program is to write to standard output. Print exactly one line containing two integers for each test case. You should output the maximum possible number of *line segment pairs* that can be picked up, followed by maximum possible score while picking up the most pairs of line segments.

The following shows sample input and output for two test cases.

Sample Input	Output for the Sample Input

2
2 2
1 2 4 2 1
1 3 4 3 2
2 1 2 4 3
3 1 3 4 4
2 2
1 2 5 2 3
7 2 6 2 3
2 3 2 1 2
3 1 3 3 1
	2 11
	1 6

The 38th Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Korea Nationwide Internet Competition

Problem J
Registration
Time Limit: 1 second

Print out your ICPC team number and team name.

Input
No input is given for this problem.

Output
Your program is to write to standard output. Print exactly two lines. The first line should contain your team number, and the second line should contain your full team name, even if your team name contains one or more blanks (a character of ASCII 32).

The following shows sample input and output, where the team number is `123` and the team name is `Your_ICPC_Team_Name`. Notice that no input is given.

Sample Input	Output for the Sample Input
	123 Your_ICPC_Team_Name

Problem J
Registration
Time Limit: 1 second

자신의 ICPC 팀 번호와 팀 이름(team name)을 그대로 출력하는 프로그램을 작성하시오.

Input

이 문제는 입력이 없다.

Output

표준출력(standard output)으로 출력해야 한다. 첫 줄에 자신의 팀 번호, 둘째 줄에 팀 이름을 출력한다. 출력할 팀 이름은 공백문자(ASCII 코드 32 번인 문자)를 포함하더라도, 공백문자를 포함하여 완전한 이름을 출력해야 한다.

다음은 팀 번호가 `123` 번, 팀 이름(team name)이 `Your_ICPC_Team_Name` 인 경우의 입출력 예제이다. 참고로 입력은 없으므로 주의한다.

Sample Input	Output for the Sample Input
	123 Your_ICPC_Team_Name

Problem K
Security
Time Limit: 2 seconds

Today most shopkeepers employ a security service. On a famous street, there are several shops. The shops are protected by a guard belonging to a security company, which is also located on the street.

For simplicity, we consider the shops and the company as points p_i on a line, running in order from left to right. The company is located at a point p_a for some a, denoted by s. Starting at the point s, the guard should visit each point p_i at least once a day to protect the shops. For each i, it takes $t[p_i, p_{i+1}]$ time for the guard to move between p_i and p_{i+1}. The latency ℓ_i of p_i is considered to be the time when the guard first visits p_i after leaving s. Obviously the latency ℓ_a of the starting point $s = p_a$ is 0. The guard shall travel to minimize the sum of latencies, which is $\sum_i \ell_i$.

For example, in Figure 1, there are six points p_1 to p_6, where the starting point s is p_3. Also $t[p_1, p_2] = 7$, $t[p_2, p_3] = 4$, $t[p_3, p_4] = 1$, $t[p_4, p_5] = 2$, $t[p_5, p_6] = 9$. When the guard first walks right from s, the latency ℓ_4 and ℓ_5 are determined as 1 and 3, respectively. If the guard travels as in Figure 1, then the sum of all the latencies is 71. It is the minimum for all possible travels of the guard.

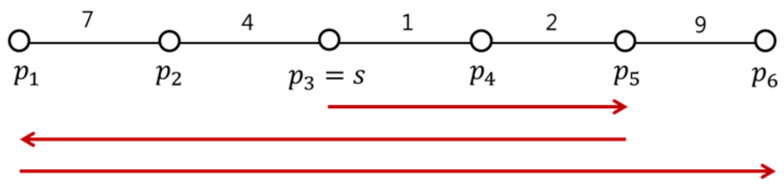

Figure 1.

Given an integer N, the number of points, and the times $t[p_i, p_{i+1}]$ for the guard to move between p_i and p_{i+1}, $i = 1, \dots, N-1$, write a program to minimize the sum of latencies of points when traveling from a starting point.

Input
Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with an integer N $(1 \le N \le 100)$, the number of points. The second line contains an integer a $(1 \le a \le N)$, saying that the a-th point is the starting point, that is, $p_a = s$. The i-th line of the following $N-1$ lines contains an integer t $(1 \le t \le 15{,}000{,}000)$, representing the time $t[p_i, p_{i+1}]$ for $i = 1, \dots, N-1$.

Output
Your program is to write to standard output. Print exactly one line for each test case. The line should contain the minimum of the sum of latencies for all possible movements of the guard.

The following shows sample input and output for two test cases.

Sample Input

```
2
6
3
7
4
1
2
9
9
5
96
24
6
2
1
3
12
48
```

Output for the Sample Input

```
71
605
```

The 38th Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Korea Nationwide Internet Competition

Problem L
Tree
Time Limit: 3 seconds

A binary tree is a basic and important data structure. As illustrated in Figure 1, a binary tree has a unique root node. Each node has at most two child nodes that are ordered from left to right. Let BT be a binary tree of n nodes. Nodes of BT have unique id numbers from 1 to n.

For BT in Figure 1, the node of id 3 is the root node. The node of id 1 has only right child node, but two nodes of id 4 and id 7 have only left child nodes. Two nodes of id 3 and id 6 have left and right child nodes. The other nodes, called leaf nodes, have no child nodes.

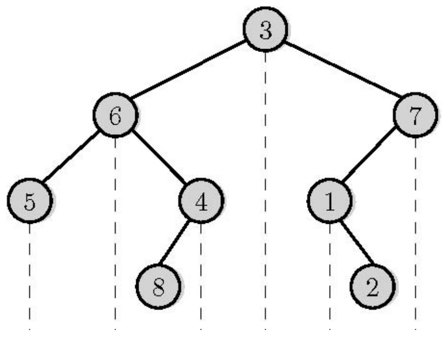

Figure 1. A binary tree with 8 nodes. The dashed vertical lines from the nodes are imaginary lines to distinguish their left and right child nodes

We have three methods to traverse all nodes of BT; preorder, inorder, and postorder traversals. These three traversals are done by the following C-style pseudo codes: For a node v of BT, we denote by $v.$left and $v.$right its left child node and right child node, respectively. If v has no left child, then $v.$left is equal to $\emptyset$. If v has no right child, then $v.$right is equal to $\emptyset$.

```
preorder(node $v$)
{
   printf(id number of $v$);
   if ($v$.left $\neq \emptyset$)
     preorder($v$.left);
   if ($v$.right $\neq \emptyset$)
     preorder($v$.right);
}
```

```
inorder(node $v$)
{
   if ($v$.left $\neq \emptyset$)
     inorder($v$.left);
   printf(id number of $v$);
   if ($v$.right $\neq \emptyset$)
     inorder($v$.right);
}
```

```
postorder(node $v$)
{
   if ($v$.left $\neq \emptyset$)
     postorder($v$.left);
   if ($v$.right $\neq \emptyset$)
     postorder($v$.right);
   printf(id number of $v$);
}
```

You are now given two lists of id numbers of BT; one is the list obtained by preorder traversal and the other is by inorder traversal, i.e., two lists obtained by calling `preorder(root node of BT)` and `inorder(root node of BT)`. It is well known that BT can be reconstructed from these two lists. You need to reconstruct BT from the preorder and inorder traversal lists of BT, and output its postorder traversal list which should be the same as the result of `postorder(root node of BT)`.

For example, if you are given preorder traversal list of 3, 6, 5, 4, 8, 7, 1, 2 and inorder traversal list of 5, 6, 8, 4, 3, 1, 2, 7, then you should reconstruct the tree as in Figure 1, and output its postorder traversal list, which is 5, 8, 4, 6, 2, 1, 7, 3.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with integer n, the number of nodes of a rooted and ordered binary tree BT, where $1 \leq n \leq 1{,}000$. Nodes of BT are assumed to have distinct id number from 1 to n. The next line contains a list of n numbers, i.e., a permutation of $\{1,2,\cdots,n\}$, which is the preorder traversal list of BT. The next line contains a list of n numbers, i.e., a permutation of $\{1,2,\cdots,n\}$, which is the inorder traversal list of the same BT. Two neighboring node id numbers in these lists have a single space between them. Note that a unique binary tree is always constructed from these preorder and inorder traversal lists.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain a permutation of n numbers of $\{1,2,\cdots,n\}$, which is the same as the postorder traversal list of BT.

The following shows sample input and output for two test cases.

Sample Input

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

Output for the Sample Input

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