After a long beer party and when ready for the check, John and his friends play the *largest sum game* and whoever wins the game will not have to pay a dime of the check. The largest sum game consists in finding the largest sum of consecutive values in a sequence of numbers; the winner is the one quickest to answer.

For example, in the sequence 23, -1, -24, 2, 23 the largest sum of consecutive values is 25 and whoever finds this value first, is the winner of the game.

Although simple (and geeky), the game is challenging because beer and arithmetic do not mix well together. However, since the group of friends are amateur programmers, each have implemented an algorithmic solution for finding the largest sum and have agreed to select the winner of the game in the form of a programming challenge: they connect their laptops to a central server that produces a random sequence of numeric values, run the solutions on this data, and the program quickest to answer wins the game.

John is tired of paying night after night without ever winning the game and he is determined to stop this situation tonight. John has hired you to write a highly efficient computer program that could beat the others in the largest sum game.

Input

The input consists of several test cases, each one defined by a line containing a sequence of N blank-separated integers $X_1, X_2, \ldots, X_N$ ($1 \leq N \leq 10^5$, $-10^3 \leq X_i \leq 10^3$ for each $1 \leq i \leq N$).

Output

For each test case, output a line with the largest sum of consecutive values in $X_1, X_2, \ldots, X_N$.

Sample Input

```
1 2 3 4 5 6 7 8 9
-1 -1 -1
23 -1 -24 2 23
1 -14 -4 14 -11 -7 6
```

Sample Output

```
45
0
25
14
```

In the country there are n cities and m bidirectional roads between them. Each city has an army. Army of the i-th city consists of a_i soldiers. Now soldiers roam. After roaming each soldier has to either stay in his city or to go to the one of neighboring cities by at moving along at most one road.

Check if is it possible that after roaming there will be exactly b_i soldiers in the i-th city.

Input

First line of input consists of two integers n and m ($1 \le n \le 100$, $0 \le m \le 200$).

Next line contains n integers $a_1, a_2, ..., a_n$ ($0 \le a_i \le 100$).

Next line contains n integers $b_1, b_2, ..., b_n$ ($0 \le b_i \le 100$).

Then m lines follow, each of them consists of two integers p and q ($1 \le p, q \le n$, $p \ne q$) denoting that there is an undirected road between cities p and q.

It is guaranteed that there is at most one road between each pair of cities.

Output

If the conditions can not be met output single word "NO".

Otherwise output word "YES" and then n lines, each of them consisting of n integers. Number in the i-th line in the j-th column should denote how many soldiers should road from city i to city j (if $i \ne j$) or how many soldiers should stay in city i (if i

$=j$).

If there are several possible answers you may output any of them.

Examples

Input
4 4 1 2 6 3 3 5 3 1 1 2 2 3 3 4 4 2
Output
YES 1 0 0 0 2 0 0 0 0 5 1 0 0 0 2 1

Input
2 0 1 2 2 1
Output
NO

Peter has n cigarettes. He smokes them one by one keeping all the butts. Out of $k > 1$ butts he can roll a new cigarette.

How many cigarettes can Peter have?

Input

Input is a sequence of lines. Each line contains two integer numbers giving the values of n and k. The input is terminated by end of file.

Output

For each line of input, output one integer number on a separate line giving the maximum number of cigarettes that Peter can have.

Sample Input

```
4 3
10 3
100 5
```

Sample Output

```
5
14
124
```

Byteland is a scarcely populated country, and residents of different cities seldom communicate with each other. There is no regular postal service and throughout most of the year a one-man courier establishment suffices to transport all freight. However, on Christmas Day there is somewhat more work for the courier than usual, and since he can only transport one parcel at a time on his bicycle, he finds himself riding back and forth among the cities of Byteland.

The courier needs to schedule a route which would allow him to leave his home city, perform the individual orders in arbitrary order (i.e. travel to the city of the sender and transport the parcel to the city of the recipient, carrying no more than one parcel at a time), and finally return home. All roads are bi-directional, but not all cities are connected by roads directly; some pairs of cities may be connected by more than one road. Knowing the lengths of all the roads and the errands to be performed, determine the length of the shortest possible cycling route for the courier.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

Each test case begins with a line containing three integers: n m b, denoting the number of cities in Byteland, the number of roads, and the number of the courier's home city, respectively (1<=n<=100,1<=b<=m<=10000). The next m lines contain three integers each, the i-th being u_i v_i d_i, which means that cities u_i and v_i are connected by a road of length d_i (1<=u_i,v_i<=100, 1<=d_i<= 10000). The following line contains integer z - the number of transport requests the courier has received (1<=z<=5). After that, z lines with the description of the orders follow. Each consists of three integers, the j-th being u_j v_j b_j, which signifies that b_j parcels should be

transported (individually) from city u_j to city v_j. The sum of all b_j does not exceed 12.

Output

For each test case output a line with a single integer - the length of the shortest possible bicycle route for the courier.

Example

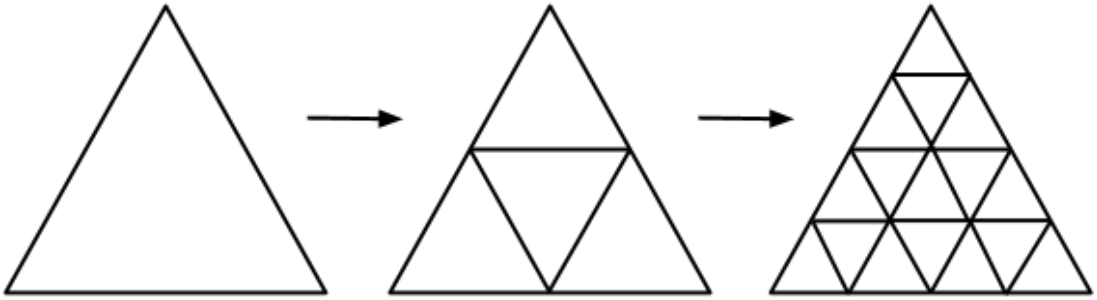
Sample input:

```
1
5 7 2
1 2 7
1 3 5
1 5 2
2 4 10
2 5 1
3 4 3
3 5 4
3
1 4 2
5 3 1
5 1 1
```

Sample output:

```
43
```

Dwarfs have planted a very interesting plant, which is a triangle directed "upwards". This plant has an amusing feature. After one year a triangle plant directed "upwards" divides into four triangle plants: three of them will point "upwards" and one will point "downwards". After another year, each triangle plant divides into four triangle plants: three of them will be directed in the same direction as the parent plant, and one of them will be directed in the opposite direction. Then each year the process repeats. The figure below illustrates this process.



Help the dwarfs find out how many triangle plants that point "upwards" will be in n years.

Input

The first line contains a single integer n ($0 \le n \le 10^{18}$) — the number of full years when the plant grew.

Please do not use the %lld specifier to read or write 64-bit integers in C++. It is preferred to use cin, cout streams or the %I64d specifier.

Output

Print a single integer — the remainder of dividing the number of plants that will point "upwards" in n years by 1000000007 ($10^9 + 7$).

Examples

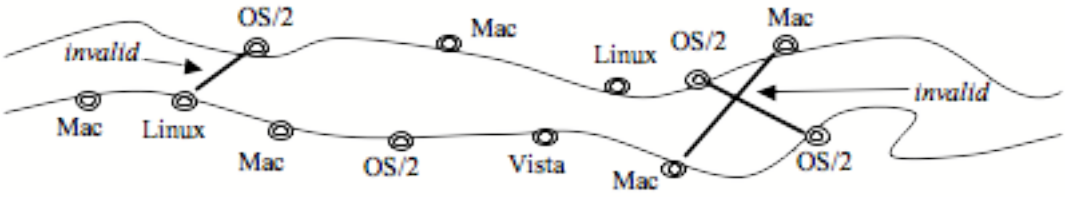
Input
1
Output
3

Input
2
Output
10

Note

The first test sample corresponds to the second triangle on the figure in the statement. The second test sample corresponds to the third one.

King Beer has a very hard region to rule, consisting of lots of cities with very sectarian operating system beliefs and high levels of trade. These cities are placed along a river, the Klsberg, along its Northern and Southern banks. The cities are economically separated from each other, since the river is wide and dangerous.



A section of the Klsberg showing some invalid bridges

King Beer would like to build some bridges connecting opposite banks of the river. He was strongly advised against making bridges between cities with different operating systems beliefs (those guys really hate each other). So, he is just going to build bridges between cities sharing the same operating system belief (even if the resulting bridges are quite long and strangely shaped). However, it is technical impossible to build bridges that cross other bridges.

The economical value of a bridge is the sum of the trade values the two cities it connects. The King wants to maximize the sum of all possible bridge values while minimizing the number of bridges to build.

Given two sets of cities, return the maximum possible sum of all bridge values and the smallest number of valid bridges necessary to achieve it.

Input

The first line is an integer with the number of samples. For each sample, the next line has a non-negative integer, not greater than 1,000, indicating the number of cities on the Northern riverbank. Then, on each line, comes the city information with the form

cityname ostype tradevalue

where, separated by empty spaces, thereare two strings, *cityname* and *ostype*, with no more than 10 characters each, and *tradevalue* which is a non-negative integer not greater than 10^6. The sequence of lines represents the cities from left to right along the riverbank. Next, there is the same kind of information to describe the Southern riverbank.

Output

For each sample, a line consisting of the maximum possible sum of all bridge values, one empty space, the number of bridges.

Sample Input

```
1
3
mordor Vista 1000000
xanadu Mac 1000
shangrila OS2 400
4
atlantis Mac 5000
hell Vista 1200
rivendell OS2 100
appleTree Mac 50
```

Sample Output

```
1002250 2
```

You have two pictures of an unusual kind of clock. The clock has n hands, each having the same length and no kind of marking whatsoever. Also, the numbers on the clock are so faded that you can't even tell anymore what direction is up in the picture. So the only thing that you see on the pictures, are n shades of the n hands, and nothing else.

You'd like to know if both images might have been taken at exactly the same time of the day, possibly with the camera rotated at different angles.

Task

Given the description of the two images, determine whether it is possible that these two pictures could be showing the same clock displaying the same time.

Input

The first line contains a single integer n ($2 \leq n \leq 200\,000$), the number of hands on the clock.

Each of the next two lines contains n integers a_i ($0 \leq a_i < 360\,000$), representing the angles of the hands of the clock on one of the images, in thousandths of a degree. The first line represents the position of the hands on the first image, whereas the second line corresponds to the second image. The number a_i denotes the angle between the recorded position of some hand and the upward direction in the image, measured clockwise. Angles of the same clock are distinct and are not given in any specific order.

Output

Output one line containing one word: `possible` if the clocks

could be showing the same time, `impossible` otherwise.

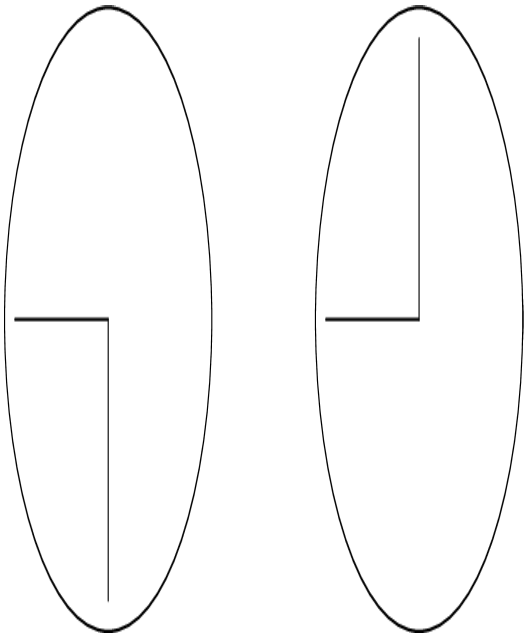


Figure 1: Sample input 2

Sample Input 1

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

Sample Output 1

```
impossible
```

Sample Input 2

```
2
0 270000
180000 270000
```

Sample Output 2

```
possible
```

Sample Input 3

```
7
140 130 110 120 125 100 105
235 205 215 220 225 200 240
```

Sample Output 3

```
impossible
```

The northern part of the Pyramid contains a very large and complicated labyrinth. The labyrinth is divided into square blocks, each of them either filled by rock, or free. There is also a little hook on the floor in the center of every free block. The ACM have found that two of the hooks must be connected by a rope that runs through the hooks in every block on the path between the connected ones. When the rope is fastened, a secret door opens. The problem is that we do not know which hooks to connect. That means also that the neccessary length of the rope is unknown. Your task is to determine the maximum length of the rope we could need for a given labyrinth.

Input

The input consists of *T* test cases. The number of them (*T*) is given on the first line of the input file. Each test case begins with a line containing two integers *C* and *R* (*3 <= C,R <= 1000*) indicating the number of columns and rows. Then exactly *R* lines follow, each containing *C* characters. These characters specify the labyrinth. Each of them is either a hash mark (#) or a period (.). Hash marks represent rocks, periods are free blocks. It is possible to walk between neighbouring blocks only, where neighbouring blocks are blocks sharing a common side. We cannot walk diagonally and we cannot step out of the labyrinth.

The labyrinth is designed in such a way that there is exactly one path between any two free blocks. Consequently, if we find the proper hooks to connect, it is easy to find the right path connecting them.

Output

Your program must print exactly one line of output for each test case. The line must contain the sentence "`Maximum rope`

length is *X*." where *X*is the length of the longest path between any two free blocks, measured in blocks.

Example

Sample Input:

```
2
3 3
###
#.#
###
7 6
#######
#.#.###
#.#.###
#.#.#.#
#.....#
#######
```

Sample output:

```
Maximum rope length is 0.
Maximum rope length is 8.
```

Warning: large Input/Output data, be careful with certain languages