Greetings From Globussoft

- Given below are 5 Programming questions, you have to solve any 3 out of 5 questions.

- These 5 questions you can attempt in any technology like C/C++, java, .Net

- To solve these 5 questions you've max. 3 hours.

- While Solving these questions you are not allowed to use any **Search Engine** like Google, Yahoo, Bing …

All the best for your test

Globussoft

QUESTION - 1

Products of a factory are packed into cylindrical boxes. All boxes have the same bases. A height of a box is a non-negative integer being a power of 2, i.e. it is equal to 2^i for some $i = 0, 1, 2, \ldots$. The number i (exponent) is called a size of a box. All boxes contain the same goods but their value may be different. Goods produced earlier are cheaper. The management decided, that the oldest (cheapest) goods should be sold out first. From the warehouse goods are transported in containers. Containers are also cylindrical. A diameter of each container is a little bigger than a diameter of a box, so that boxes can be easily put into containers. A height of a container is a non-negative power of 2. This number is called a size of a container. For safe transport containers should be tightly packed with boxes, i.e. the sum of heights of boxes placed in a container have to be equal to the height of this container. A set of containers was delivered to the warehouse. Check if it is possible to pack all the containers tight with boxes that are currently stored in the warehouse. If so, find the minimal value of goods that can be tightly packed into these containers.

Consider a warehouse with 5 boxes. Their sizes and values of their contents are given below:

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

Two containers of size 1 and 2 can be tightly packed with two boxes of total value 3, 4 or 5, or three boxes with total value 9. The container of size 5 cannot be tightly packed with boxes from the warehouse.

Task

Write a program that for each test case:

- reads descriptions of boxes (size, value) from a warehouse and descriptions of containers (how many containers of a given size we have);
- checks if all containers can be tightly packed with boxes from the warehouse and if so, computes the minimal value of goods that can be tightly packed into these containers;
- writes the result.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line.

In the first line of a test case there is an integer n, $1 <= n <= 10000$, which is the number of boxes in the warehouse. In each of the following n lines there are written two non-negative integers

separated by a single space. These numbers describe a single box. First of them is the size of the box and the second - the value of goods contained in this box. The size is not greater than 1000 and the value is not greater than 10000. The next line contains a positive integer *q*, which is the number of different sizes of containers delivered to the warehouse. In each of the following *q* lines there are two positive integers separated by a single space. The first integer is the size of a container and the second one is the number of containers of this size. The maximal number of containers is 5000, a size of a container is not greater than 1000.

Output

For each test case your program should output exactly one line containing:

- a single word "No" if it is not possible to pack the containers from the given set tight with the boxes from the warehouse, or
- a single integer equal to the minimal value of goods in boxes with which all the containers from the given set can be packed tight.

Example

Sample input:

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

Sample output:

```
3
```

QUESTION – 2

A manufacturer of sweets has started production of a new type of sweet called *rock*. Rock comes in sticks composed of one-centimetre-long segments, some of which are sweet, and the rest are sour. Before sale, the rock is broken up into smaller pieces by splitting it at the connections of some segments.

Today's children are very particular about what they eat, and they will only buy a piece of rock if it contains more sweet segments than sour ones. Try to determine the total length of rock which can be sold after breaking up the rock in the best possible way.

Input

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

For each test case, the first line of input contains one integer N - the length of the stick in centimetres (1<=N<=200). The next line is a sequence of N characters '0' or '1', describing the segments of the stick from the left end to the right end ('0' denotes a sour segment, '1' - a sweet one).

Output

For each test case output a line with a single integer: the total length of rock that can be sold after breaking up the rock in the best possible way.

Example

Sample input:

```
2
15
100110001010001
16
0010111101100000
```

Sample output:

```
9
13
```

QUESTION – 3

You are given scales for weighing loads. On the left side lies a single stone of known weight $W<2^N$. You own a set of N different weights, weighing 1, 2, 4, ..., 2^{N-1} units of mass respectively. Determine how many possible ways there are of placing some weights on the sides of the scales, so as to balance them (put them in a state of equilibrium). Output this value modulo a small integer D.

Input

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

For each test case, the first line contains three integers: N L D, where N denotes the number of weights at your disposal, L is the length of the binary representation of number W, and D is the modulus (1<= L<= N<= 1000000, 2<= D<=100). The second line contains the value of W,

encoded in the binary system as a sequence of exactly L characters 0 or 1 without separating spaces.

Output

For each test case, output a single line containing one integer - the calculated number of possible weight placements, modulo D.

Example

Sample input:

```
2
6 4 6
1000
6 6 100
100110
```

Sample output:

```
3
5
```

QUESTION – 4

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
```

QUESTION – 5

After trying to solve Problem Number 31 (Fast Multiplication) with some script languages that support arbitrary large integers and timing out, you wonder what would be the best language to do fast multiplication of integers. And naturally it comes to your mind: Of course it is brainf**k, because there are only very cheap operations in that language.

Input

Two positive integers, ended with a line feed (ASCII 10) each.

Output

The product of the two integers, terminated by a line feed. You may assume that this number will be less than 10000.

Example

```
Input:
1
2

Output:
2
```