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, PHP

- To solve these 3 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

Write a program that checks if the given logical expression is a tautology. The logical expression is a tautology if it is always true, regardless of logical value of its variables.

Input

On the first line there is the number of expressions to check (at most 35). The expression is in a prefix notation, that means that operator precedes its arguments. The following logical operators will be used:

```
C - and
D - or
I - implies
E - if, and only if
N - not
```

The variables will be lowercase letters (a-z). There will be no more than 16 different letters in the expression. The length of the expression will not exceed 111 characters.

Output

For each expression write one word: YES if it is a tautology, NO in other case.

Example

Sample input:

```
7
IIpqDpNp
NCNpp
Iaz
NNNNNNNp
IIqrIIpqIpr
Ipp
Ezz
```

Sample output:

```
YES
YES
NO
NO
YES
YES
YES
```

QUESTION – 2

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 – 3

Every member of Byteland Credit Society is entitled to loan any amount of Bytelandish ducats unless it is 10^{30} or more, but he has to return the whole amount within seven days. There are 100 ATMs in the Client Service Room of the Society. They are numbered from 0 to 99. Every ATM can perform one action only: it can pay or receive a fixed amount. The i-th ATM pays 2^i ducats if i is even or it receives 2^i ducats if i is odd. If a client is going to loan a fixed sum of money it is necessary to check if he is able to get the money using every ATM at most once. If so, numbers of ATMs he has to use should be determined. It is also necessary to check if the client can return the money in a similar way, and if so, to determine numbers of ATMs he has to use.

Example

A client who is going to loan 7 ducats gets 16 ducats from the ATM # 4 and 1 ducat from the ATM # 0 and then he returns 8 ducats in the ATM # 3 and 2 ducats in the ATM # 1. In order to return the amount of 7 ducats he receives 1 ducat from the ATM # 0 and then he returns 8 ducats in ATM # 3.

Task

Write a program that:

- reads the number of clients *n*, for every client reads from the same file the amount of money he is going to loan;
- checks for every client if he is able to get the money using every ATM at most once and if so, determines the numbers of ATMs he has to use;
- outputs the results.

Input

In the first line of input there is one positive integer $n <= 10000$, which equals the number of clients.

In each of the following *n* lines there is one positive integer less than 10^{30} (at most 30 decimal digits). The number in the *i*-th line describes the amount of ducats which the client *i* is going to loan.

Output

For each client you should output two lines with a decreasing sequence of positive integers from the range [0..99] separated by single spaces, or one word "No":

In the first line of the *i*-th pair of lines there should be numbers of ATMs (in decreasing order) that the client *i* should use to get his loan or one word "No" if the loan cannot be received according to the rules;

In the second line of the *i*-th pair there should be numbers of ATMs (in decreasing order) which the client *i* should use to return his loan or the word "No".

Example

Sample input:

```
2
7
633825300114114700748351602698
```

Sample output:

```
4 3 1 0
3 0
No
99 3 1
```

QUESTION – 4

Two players take part in the game **polygons**. A convex polygon with *n* vertices divided by *n-3* diagonals into *n-2* triangles is necessary. These diagonals may intersect in vertices of the polygon only. One of the triangles is black and the remaining ones are white. Players proceed in alternate turns. Each player, when its turn comes, cuts away one triangle from the polygon. players are allowed to cut off triangles along the given diagonals. The winner is the player who cuts away the black triangle.
NOTE: We call a polygon **convex** if a segment joining any two points of the polygon is contained in the polygon.

Task

Write a program which:

- reads from the standard input the description of the polygon,
- verifies whether the player who starts the game has a winning strategy,
- writes the result to the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. The first line of each test case contains an integer *n*, *4 <= n <= 50000*. This is the number of vertices in the polygon. The vertices of the polygon are numbered, clockwise, from *0* to *n-1*.
The next *n-2* lines comprise descriptions of triangles in the polygon. In the *i+1*-th line, *1 <= i <= n-2*, there are three non-negative integers *a, b, c* separated by single spaces. Theses are numbers of vertices of the *i*-th triangle. The first triangle in a sequence is black.

Output

The output for each test case should have one line with the word:

- YES, if the player, who starts the game has a winning strategy,
- NO, if he does not have a winning strategy.

Example

Sample input:

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

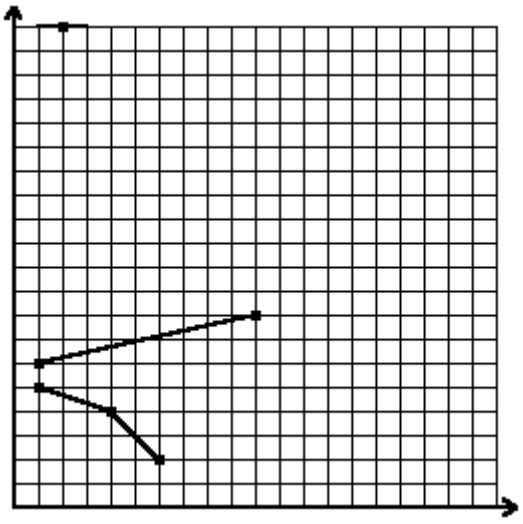
Sample output:

```
YES
```

QUESTION – 5

We have a cartesian coordinate system drawn on a sheet of paper. Let us consider broken lines that can be drawn with a single pencil stroke from the left to the right side of the sheet. We also require that for each segment of the line the angle between the straight line containing this segment and the OX axis belongs to [-45°, 45°] range. A broken line fulfilling above conditions is called a flat broken line. Suppose we are given *n* distinct points with integer coordinates. What is the minimal number of flat broken lines that should be drawn in order to cover all the points (a point is covered by a line if it belongs to this line)?

Example



For 6 points whose coordinates are (1,6), (10,8), (1,5), (2,20), (4,4), (6,2) the minimal number of flat broken lines covering them is 3.

Task

Write a program that for each test case:

- reads the number of points and their coordinates;
- computes the minimal number of flat broken lines that should be drawn to cover all the points;
- outputs 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 one positive integer *n*, not greater than 30000, which denotes the number of points. In the following *n* lines there are coordinates of points. Each line contains two integers *x*, *y* separated by a single space, 0 <= *x* <= 30000, 0 <= *y* <= 30000. The numbers in the *i*-th line are the coordinates of the *i*-th point.

Output

For each test case you should output one line with the minimal number of flat broken lines that should be drawn to cover all the points.

Example

Sample input:

```
1
6
1 6
10 8
```

```
1 5
2 20
4 4
6 2
```

Sample output:

```
3
```