CMU ACM ICPC Team Selection Round 2
September 23, 2015

A. Queries

time limit per test: 0.25 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

Mathematicians are interesting (sometimes, I would say, even crazy) people. For example, my friend, a mathematician, thinks that it is very fun to play with a sequence of integer numbers. He writes the sequence in a row. If he wants he increases one number of the sequence, sometimes it is more interesting to decrease it (do you know why?..) And he likes to add the numbers in the interval $[l;r]$. But showing that he is really cool he adds only numbers which are equal some mod (modulo m).

Guess what he asked me, when he knew that I am a programmer? Yep, indeed, he asked me to write a program which could process these queries (n is the length of the sequence):

- `+ p r` It increases the number with index `p` by `r`. ($1 \leq \mathbf{p} \leq n, 1 \leq \mathbf{r} \leq 1000000000$)
  You have to output the number after the increase.
- `- p r` It decreases the number with index `p` by `r`. ($1 \leq \mathbf{p} \leq n, 1 \leq \mathbf{r} \leq 1000000000$) You must not decrease the number if it would become negative.
  You have to output the number after the decrease.
- `s l r mod` You have to output the sum of numbers in the interval $[\mathbf{l};\mathbf{r}]$ which are equal `mod` (modulo m). ($1 \leq \mathbf{l} \leq \mathbf{r} \leq n$) ($0 \leq \mathbf{mod} < m$)

Input

The first line of each test case contains the number of elements of the sequence n and the number m. ($1 \leq n \leq 10000$) ($1 \leq m \leq 10$)

The second line contains n initial numbers of the sequence. ($0 \leq number \leq 1000000000$)

The third line of each test case contains the number of queries q ($1 \leq q \leq 10000$).

The following q lines contains the queries (one query per line).

Output

Output q lines - the answers to the queries.

Sample test(s)

input

```
3 4
1 2 3
3
s 1 3 2
+ 2 1
- 1 2
```

output

```
2
3
1
```

B. Matrix

time limit per test: 0.5 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

You're given a matrix with n rows and n columns. A basic property of a square matrix is the main diagonal: all cells that have the same row and column number.

We consider all diagonals that are parallel to the main one. We consider them from left-low corner to the right-upper one. So, the first cell of each diagonal will be, in order: (n, 1) (n - 1, 1) ... (1, 1) (1, 2) ... (1, n).

You need to choose one number from each diagonal. More, all 2*n-1 numbers must be pairwise distinct.

Input

The first line contains number n $(1 \leq n \leq 300)$. Next n lines contain n numbers, representing the elements of the matrix. All elements of the matrix are between 1 and 10^9.

Output

If there is no solution, output "NO". Otherwise, output "YES". Next, on the same line, output 2n-1 numbers, separated by spaces. Each number represents the chosen value from a diagonal. Diagonals are considered in the order given by the problem.

Sample test(s)

input

```
2
1 1
1 1
```

output

```
NO
```

C. Fitting boxes

time limit per test: 0.25 seconds

memory limit per test: 64 megabytes

input: standard input

output: standard output

Programming is fun, but programmers usually don't like geometry problems, though they are often part of the programming competitions. Jonas participated in a programming competition yesterday and one of the problems that he hadn't solved was a geometry problem. Jonas is a very stubborn programmer, and he always works hard to solve the problems he didn't solve in a contest. Certainly, Jonas was able to solve the problem when he got home after a while, but let's see if you can do it better than Jonas today. In this problem you are given 2 rectangles and you need to find if you can fit one of them into the other. The dimensions $(A1, B1)$, $(A2, B2)$ of the rectangles are given. Rectangles can be rotated, if necessary, in order to fit them. The width of the sides of the rectangles is negligible. The rectangle inside can coincide with the outer rectangle sides, for example 1 x 1 rectangle fits inside 2 x 1.

Input

Each test case contains 4 positive integers separated by a space character, $A1 \le 10^3$, $B1 \le 10^3$, $A2 \le 10^3$, $B2 \le 10^3$.

Output

Output "Yes" if it's possible to fit one box into another, or "No" if it's not possible.

Sample test(s)

input
```
4 4 4 4
```
output
```
Yes
```

input
```
20 1 1 10
```
output
```
Yes
```

input
```
10 10 5 16
```
output
```
No
```

input
```
10 10 11 1
```
output
```
Yes
```

input
```
100 100 200 1
```
output
```
No
```

Problem D. Jingles of a String

Input file:	`jingles.in`
Output file:	`jingles.out`
Time limit:	2 seconds
Memory limit:	512 megabytes

Given a string s a *jingle* $J(L, R)$ of its substring $s[L..R]$ is the set of characters that appear among its characters. For example, if $s =$ "`abacaba`", then $s[3..5] =$ "`aca`" and $J(3, 5) = \{a, c\}$.

You are given a string s. Find all possible jingles of its non-empty substrings and for each possible jingle find the longest substring of s with such jingle.

In the example above, there are 6 sets that appear as jingles of substrings of s. For example, for a set $\{a, b\}$ there are six substrings that has such jingle: $s[1..2]$, $s[1..3]$, $s[2..3]$, $s[5..6]$, $s[5..7]$, $s[6..7]$. Among them $s[1..3]$ (as well as $s[5..7]$) has length 3, this is the longest substring of s with such jingle.

Since printing all jingles would make output file too large, you have to output the sum

$$v(s) = \sum_{J \in \mathbb{J}} S(J)L(J).$$

Here $\mathbb{J}$ is the set of all jingles of the string, $S(J)$ is the number of characters in J, $L(J)$ is the length of longest substring of s that has J as its jingle.

Input

The input file contains multiple test cases.

The first line of the input file contains t — the number of test cases.

Each of the following t lines contains a string s that consists of at most 100 000 lowercase characters.

The sum of lengths of strings for all test cases in the input file doesn't exceed 100 000.

Output

For each test case first output two integers: d — the number of different sets that appear as jingles of some substrings of s and $v(s)$ — the sum described in the problem statement.

Examples

jingles.in	jingles.out
2 abacaba abbcccdddd	6 36 10 125

Problem E. Elegant Square

Input file:	`elegant.in`
Output file:	`elegant.out`
Time limit:	2 seconds
Memory limit:	512 megabytes

Many people know about magic squares — squares that contain distinct numbers and have equals sums of rows and columns. Recently Eve has heard about magic squares, and now she has invented her own version: *elegant squares*.

Eve calls a square of $n \times n$ integers elegant if the following conditions are satisfied:

- All entries of the square are distinct positive integers.
- All integers are square free. That means that no integer is divisible by t^2 for any $t > 1$.
- The product of numbers in any row and any column is the same.

For example, the picture below shows an elegant 3×3 square.

$$\begin{array}{ccc} 1 & 21 & 10 \\ 6 & 5 & 7 \\ 35 & 2 & 3 \end{array}$$

All of its entries are distinct positive square free integers, and product of any row and any column is 210.

Help Eve, find an $n \times n$ elegant square. All numbers in the square must not exceed 10^{18}. It is guaranteed that for the given constraints there exists such square.

Input

The input file a single integer n ($3 \le n \le 30$).

Output

Output $n \times n$ integers: the found elegant square. All printed integers must not exceed 10^{18}.

Examples

elegant.in	elegant.out

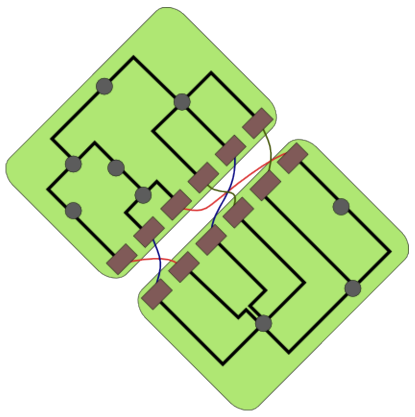
3	1 21 10
	6 5 7
	35 2 3

Problem F. Graphic Madness

In Byteland, there are two leading video card manufacturers: Bitotronics and 3D-Bytes. Their top-of-the-line cards are quite similar. Each of them consists of many *nodes*, connected with wires transferring the signal that is being processed. The products contain two kinds of nodes: sockets and processors. The wire network fulfills the following conditions:

- Each socket is connected to exactly one processor and no other sockets.
- Each processor is connected to at least two other nodes.
- For any two nodes in the network, there is exactly one path of wires connecting them. In other words, the graph of connections between nodes is a tree.

Bitthew loves to tinker with computer hardware. He has bought two video cards, one from each manufacturer. Since accidentally the cards have the same number of sockets, he has decided to connect each socket of the Bitotronics card to a distinct socket of the 3D-Bytes card with cables. The device he obtained looks like this:



Bitthew would like to squeeze out maximum processing power from the device. In order to do that, he wants to find a path through wires and cables that the processed signal can take. The path should visit each node of both cards exactly once, and it should start and end at the same node on the same card. Help Bitthew find out whether this can be done.

Input

The first line of the input contains the number of test cases T. The descriptions of the test cases follow:

Each test case starts with three integers k, n, m $(2 \leqslant k \leqslant 1\,000, 1 \leqslant n, m \leqslant 1\,000)$ denoting respectively the number of sockets on each card, the number of processors on the Bitotronics card, and the number of processors on the 3D-Bytes card. The nodes on the cards are named as follows:

- the sockets on the Bitotronics card: $\text{AS}1, \text{AS}2, \ldots, \text{AS}k$
- the processors on the Bitotronics card: $\text{AP}1, \text{AP}2, \ldots, \text{AP}n$

- the sockets on the 3D-Bytes card: $\mathrm{BS}1, \mathrm{BS}2, \ldots, \mathrm{BS}k$
- the processors on the 3D-Bytes card: $\mathrm{BP}1, \mathrm{BP}2, \ldots, \mathrm{BP}m$

The next $n + k - 1$ lines contain the description of the wire network on the Bitotronics card. Each of these lines contains the names of two different nodes on that card that are connected directly by a wire. The description is followed by a blank line. The next $m + k - 1$ lines contain the description of the wire network on the 3D-Bytes card in the same format. The description is followed by another blank line. The last k lines of the test case describe the cables added by Bitthew. Each of these lines contains the names of two sockets on different cards that are directly connected by a cable. Every socket will be present on exactly one of these k lines. There is a blank line after each test case except the last one.

Output

Print the answers to the test cases in the order in which they appear in the input. For each test case print a single line containing the answer. If there exists no path with the desired properties, output **NO**. Otherwise, output **YES** followed by a description of the path: $n+m+2k$ distinct nodes in the order in which the signal will pass through them. Every two consecutive nodes should be connected by a wire or a cable. Additionally, the first and the last node must be connected.

Example

Input	Output (should be a single line)

```
1
2 1 11
AS1 AP1
AS2 AP1

BS1 BP1
BS2 BP11
BP1 BP2
BP2 BP3
BP3 BP4
BP4 BP5
BP5 BP6
BP6 BP7
BP7 BP8
BP8 BP9
BP9 BP10
BP10 BP11

AS1 BS2
BS1 AS2
```
	```
	YES BP11 BP10 BP9 BP8 BP7 BP6 BP5 BP4
	BP3 BP2 BP1 BS1 AS2 AP1 AS1 BS2
	```

Problem G. Kingdoms

Several kingdoms got into serious financial troubles. For many years, they have been secretly borrowing more and more money from each other. Now, with their liabilities exposed, the crash is inevitable...

There are n kingdoms. For each pair (A, B) of kingdoms, the amount of gold that kingdom A owes to kingdom B is expressed by an integer number d_{AB} (we assume that $d_{BA} = -d_{AB}$). If a kingdom has negative balance (has to pay more than it can receive), it may bankrupt. Bankruptcy removes all liabilities, both positive and negative, as if the kingdom ceased to exist. The next kingdom may then bankrupt, and so on, until all remaining kingdoms are financially stable.

Depending on who falls first, different scenarios may occur—in particular, sometimes only one kingdom might remain. Determine, for every kingdom, whether it can become the only survivor.

Input

The first line of the input contains the number of test cases T. The descriptions of the test cases follow:

The description of each test case starts with a line containing the number of the kingdoms n, $1 \leqslant n \leqslant 20$. Then n lines follow, each containing n space-separated numbers. The j-th number in the i-th line is the number d_{ij} of gold coins that the i-th kingdom owes to the j-th one. You may assume that $d_{ii} = 0$ and $d_{ij} = -d_{ji}$ for every $1 \leqslant i, j \leqslant n$. Also, $|d_{ij}| \leqslant 10^6$ for all possible i, j.

Output

Print the answers to the test cases in the order in which they appear in the input. For each test case, print a single line containing the indices of the kingdoms that can become the sole survivors, in increasing order. If there are no such kingdoms, print a single number **0**.

Example

Input	Output

1
3
0 -3 1
3 0 -2
-1 2 0
	
	1 3

H. Boredom

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alex doesn't like boredom. That's why whenever he gets bored, he comes up with games. One long winter evening he came up with a game and decided to play it.

Given a sequence a consisting of n integers. The player can make several steps. In a single step he can choose an element of the sequence (let's denote it a_k) and delete it, at that all elements equal to $a_k + 1$ and a_k - 1 also must be deleted from the sequence. That step brings a_k points to the player.

Alex is a perfectionist, so he decided to get as many points as possible. Help him.

Input

The first line contains integer n ($1 \le n \le 10^5$) that shows how many numbers are in Alex's sequence.

The second line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le 10^5$).

Output

Print a single integer — the maximum number of points that Alex can earn.

Sample test(s)

input

```
2
1 2
```

output

```
2
```

input

```
3
1 2 3
```

output

```
4
```

input

```
9
1 2 1 3 2 2 2 2 3
```

output

```
10
```

Note

Consider the third test example. At first step we need to choose any element equal to 2. After that step our sequence looks like this $[2, 2, 2, 2]$. Then we do 4 steps, on each step we choose any element equals to 2. In total we earn 10 points.