A. Odd Selection

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

Shubham has an array a of size n, and wants to select exactly x elements from it, such that their sum is odd. These elements do not have to be consecutive. The elements of the array are not guaranteed to be distinct.

Tell him whether he can do so.

Input

The first line of the input contains a single integer t $(1 \le t \le 100)$ — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n and x $(1 \le x \le n \le 1000)$ — the length of the array and the number of elements you need to choose.

The next line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 1000)$ — elements of the array.

Output

For each test case, print "Yes" or "No" depending on whether it is possible to choose x elements such that their sum is odd.

You may print every letter in any case you want.

Example

input

```
5
1 1
999
1 1
1000
2 1
51 50
2 2
51 50
3 3
101 102 103
```

output

```
Yes
No
Yes
Yes
No
```

Note

For 1st case: We must select element 999, and the sum is odd.

For 2nd case: We must select element 1000, so overall sum is not odd.

For 3rd case: We can select element 51.

For 4th case: We must select both elements 50 and 51 — so overall sum is odd.

For 5th case: We must select all elements — but overall sum is not odd.

B. Subsequence Hate

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

Shubham has a binary string s. A binary string is a string containing only characters "0" and "1".

He can perform the following operation on the string any amount of times:

- Select an index of the string, and flip the character at that index. This means, if the character was "0", it becomes "1", and vice versa.

A string is called good if it does not contain "010" or "101" as a subsequence — for instance, "1001" contains "101" as a subsequence, hence it is not a good string, while "1000" doesn't contain neither "010" nor "101" as subsequences, so it is a good string.

What is the minimum number of operations he will have to perform, so that the string becomes good? It can be shown that with these operations we can make any string good.

A string a is a subsequence of a string b if a can be obtained from b by deletion of several (possibly, zero or all) characters.

Input

The first line of the input contains a single integer t $(1 \le t \le 100)$ — the number of test cases.

Each of the next t lines contains a binary string s $(1 \le |s| \le 1000)$.

Output

For every string, output the minimum number of operations required to make it good.

Example

input

```
7
001
100
101
010
0
1
001100
```

output

```
0
0
1
1
0
0
2
```

Note

In test cases $1, 2, 5, 6$ no operations are required since they are already good strings.

For the 3rd test case: "001" can be achieved by flipping the first character — and is one of the possible ways to get a good string.

For the 4th test case: "000" can be achieved by flipping the second character — and is one of the possible ways to get a good string.

For the 7th test case: "000000" can be achieved by flipping the third and fourth characters — and is one of the possible ways to get a good string.

C. Game On Leaves

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Ayush and Ashish play a game on an unrooted tree consisting of n nodes numbered 1 to n. Players make the following move in turns:

- Select any leaf node in the tree and remove it together with any edge which has this node as one of its endpoints. A leaf node is a node with degree less than or equal to 1.

A tree is a connected undirected graph without cycles.

There is a special node numbered x. The player who removes this node wins the game.

Ayush moves first. Determine the winner of the game if each player plays optimally.

Input

The first line of the input contains a single integer t $(1 \le t \le 10)$ — the number of testcases. The description of the test cases follows.

The first line of each testcase contains two integers n and x $(1 \le n \le 1000, 1 \le x \le n)$ — the number of nodes in the tree and the special node respectively.

Each of the next $n-1$ lines contain two integers u, v $(1 \le u, v \le n,\ u \ne v)$, meaning that there is an edge between nodes u and v in the tree.

Output

For every test case, if Ayush wins the game, print "Ayush", otherwise print "Ashish" (without quotes).

Examples

input

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

output

```
Ashish
```

input

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

output

```
Ayush
```

Note

For the 1st test case, Ayush can only remove node 2 or 3, after which node 1 becomes a leaf node and Ashish can remove it in his turn.

For the 2nd test case, Ayush can remove node 2 in the first move itself.

D. Guess The Maximums

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is an interactive problem.

Ayush devised a new scheme to set the password of his lock. The lock has k slots where each slot can hold integers from 1 to n. The password P is a sequence of k integers each in the range $[1, n]$, i-th element of which goes into the i-th slot of the lock.

To set the password of his lock, Ayush comes up with an array A of n integers each in the range $[1, n]$ (not necessarily distinct). He then picks k **non-empty mutually disjoint** subsets of indices $S_1, S_2, \ldots, S_k$ $(S_i \underset{i \neq j}{\cap} S_j = \emptyset)$ and sets his password as $P_i = \max\limits_{j \notin S_i} A[j]$. In other words, the i-th integer in the password is equal to the maximum over all elements of A whose indices do not belong to S_i.

You are given the subsets of indices chosen by Ayush. You need to guess the password. To make a query, you can choose a non-empty subset of indices of the array and ask the maximum of all elements of the array with index in this subset. **You can ask no more than 12 queries**.

Input

The first line of the input contains a single integer t $(1 \leq t \leq 10)$ — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n and k $(2 \leq n \leq 1000, 1 \leq k \leq n)$ — the size of the array and the number of subsets. k lines follow. The i-th line contains an integer c $(1 \leq c < n)$ — the size of subset S_i, followed by c distinct integers in the range $[1, n]$ — indices from the subset S_i.

It is guaranteed that the intersection of any two subsets is empty.

Interaction

To ask a query print a single line:

- In the beginning print "`? c` " (without quotes) where c $(1 \leq c \leq n)$ denotes the size of the subset of indices being queried, followed by c **distinct** space-separated integers in the range $[1, n]$.

For each query, you will receive an integer x — the maximum of value in the array among all the indices queried. If the subset of indices queried is invalid or you exceeded the number of queries (for example one of the indices is greater than n) then you will get $x = -1$. In this case, you should terminate the program immediately.

When you have guessed the password, print a single line "`! `" (without quotes), followed by k space-separated integers — the password sequence.

Guessing the password does **not** count towards the number of queries asked.

After this, you should read a string. If you guess the password correctly, you will receive the string "`Correct`". In this case, you should continue solving the remaining test cases. If the guessed password is incorrect, you will receive the string "`Incorrect`". In this case, you should terminate the program immediately.

The interactor is not adaptive. The array A does not change with queries.

After printing a query do not forget to output end of line and flush the output. Otherwise, you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages.

Hacks

To hack the solution use the following test format:

The first line of the input should contain a single integer t $(1 \leq t \leq 10)$ — the number of test cases.

The first line of each test case should contain two integers n and k $(2 \leq n \leq 1000, 1 \leq k \leq n)$ — the size of the array and the number of subsets. The next line should consist of n space separated integers in the range $[1, n]$ — the array A. k lines should follow. The i-th line should contain an integer c $(1 \leq c < n)$ — the size of subset S_i, followed by c distinct integers in the range $[1, n]$ — indices from the subset S_i.

The intersection of any two subsets has to be empty.

Example

input

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

1

2

3

4

Correct
```

output

```
? 1 1

? 1 2

? 1 3

? 1 4

! 4 3
```

Note

The array A in the example is $[1, 2, 3, 4]$. The length of the password is 2. The first element of the password is the maximum of $A[2]$, $A[4]$ (since the first subset contains indices 1 and 3, we take maximum over remaining indices). The second element of the password is the maximum of $A[1]$, $A[3]$ (since the second subset contains indices 2, 4).

Do not forget to read the string "`Correct`" / "`Incorrect`" after guessing the password.

E. Tree Shuffling

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Ashish has a tree consisting of n nodes numbered 1 to n rooted at node 1. The i-th node in the tree has a cost a_i, and binary digit b_i is written in it. He wants to have binary digit c_i written in the i-th node in the end.

To achieve this, he can perform the following operation any number of times:

- Select any k nodes from the subtree of any node u, and shuffle the digits in these nodes as he wishes, incurring a cost of $k \cdot a_u$. Here, he can choose k ranging from 1 to the size of the subtree of u.

He wants to perform the operations in such a way that every node finally has the digit corresponding to its target.

Help him find the minimum total cost he needs to spend so that after all the operations, every node u has digit c_u written in it, or determine that it is impossible.

Input

First line contains a single integer n $(1 \leq n \leq 2 \cdot 10^5)$ denoting the number of nodes in the tree.

i-th line of the next n lines contains 3 space-separated integers a_i, b_i, c_i $(1 \le a_i \le 10^9, 0 \le b_i, c_i \le 1)$ — the cost of the i-th node, its initial digit and its goal digit.

Each of the next $n - 1$ lines contain two integers u, v $(1 \le u, v \le n,\ u \ne v)$, meaning that there is an edge between nodes u and v in the tree.

Output

Print the minimum total cost to make every node reach its target digit, and -1 if it is impossible.

Examples

input

```
5
1 0 1
20 1 0
300 0 1
4000 0 0
50000 1 0
1 2
2 3
2 4
1 5
```

output

```
4
```

input

```
5
10000 0 1
2000 1 0
300 0 1
40 0 0
1 1 0
1 2
2 3
2 4
1 5
```

output

```
24000
```

input

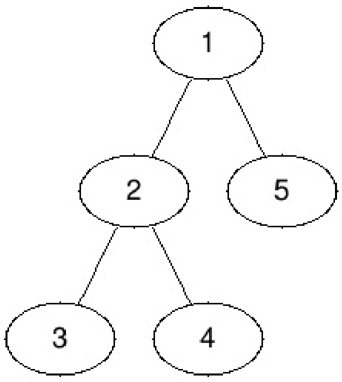
```
2
109 0 1
205 0 1
1 2
```

output

```
-1
```

Note

The tree corresponding to samples 1 and 2 are:



In sample 1, we can choose node 1 and $k = 4$ for a cost of $4 \cdot 1 = 4$ and select nodes $1, 2, 3, 5$, shuffle their digits and get the desired digits in every node.

In sample 2, we can choose node 1 and $k = 2$ for a cost of $10000 \cdot 2$, select nodes $1, 5$ and exchange their digits, and similarly, choose node 2 and $k = 2$ for a cost of $2000 \cdot 2$, select nodes $2, 3$ and exchange their digits to get the desired digits in every node.

In sample 3, it is impossible to get the desired digits, because there is no node with digit 1 initially.

F. Rotating Substrings

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input

output: standard output

You are given two strings s and t, each of length n and consisting of lowercase Latin alphabets. You want to make s equal to t.

You can perform the following operation on s any number of times to achieve it —

- Choose any substring of s and rotate it clockwise once, that is, if the selected substring is $s[l, l+1...r]$, then it becomes $s[r, l, l+1 \ldots r-1]$. All the remaining characters of s stay in their position.
  For example, on rotating the substring $[2, 4]$, string "abcde" becomes "adbce".

A string a is a substring of a string b if a can be obtained from b by deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

Find the minimum number of operations required to convert s to t, or determine that it's impossible.

Input

The first line of the input contains a single integer t $(1 \leq t \leq 2000)$ — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n $(1 \leq n \leq 2000)$ — the length of the strings.

The second and the third lines contain strings s and t respectively.

The sum of n over all the test cases does not exceed 2000.

Output

For each test case, output the minimum number of operations to convert s to t. If it is not possible to convert s to t, output -1 instead.

Example

input

```
6
1
a
a
2
ab
ba
3
abc
cab
3
abc
cba
4
abab
baba
4
abcc
aabc
```

output

```
0
1
1
2
1
-1
```

Note

For the 1-st test case, since s and t are equal, you don't need to apply any operation.

For the 2-nd test case, you only need to apply one operation on the entire string ab to convert it to ba.

For the 3-rd test case, you only need to apply one operation on the entire string abc to convert it to cab.

For the 4-th test case, you need to apply the operation twice: first on the entire string abc to convert it to cab and then on the substring of length 2 beginning at the second character to convert it to cba.

For the 5-th test case, you only need to apply one operation on the entire string abab to convert it to baba.

For the 6-th test case, it is not possible to convert string s to t.