Problem A Art Museum

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	64 megabytes

EPFL (Extreme Programmers For Life) want to build their 57th art museum. This museum would be better, bigger and simply more amazing than the last 56 museums. This would make EPFL great again.

To achieve this, EPFL will not only require the help of top-notch Japanese architects, but also of all Martians. However, the Martians are spread across several galaxies and therefore have different time availabilities. You thought the time difference between Europe and USA was annoying?

Help EPFL find the maximum number of galaxies that are available at the same hour?

Input

The first line contains the integer n, the number of galaxies ($1 \leq n \leq 10^5$).

On each of the next n lines, there will be two space-separated integers, a and b ($0 \leq a < b \leq 24$).

This means that the Martians of this galaxy are available from the beginning of the a-th hour to the beginning of the b-th hour.

Output

Print the maximum number of galaxies available in any timeslot of one hour.

Examples

standard input	standard output

2
1 14
14 21
	1

7
9 16
16 18
5 12
15 24
9 20
20 22
23 24
	3

Note

In the first example, the Martians of the first galaxy are available from 01:00 to 14:00, and the Martians of the second galaxy are available from 14:00 to 21:00. Therefore, at any given hour, the maximum number of available galaxies is 1.

In the second example, the timeslots in which 3 galaxies are available are: 9:00 - 12:00, 15:00 - 16:00, and 16:00 - 18:00. No one-hour timeslot has more than 3 galaxies available. Therefore, the answer is 3.

Problem B Secret Passage

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 megabytes

The Martians have invaded EPFL. Five minutes ago, they were at Esplanade and started heading towards the EPFL metro station to destroy it so that no student can return home (no one likes taking the bus anyway, and walking is just too hard!).

You are currently at Esplanade, and you need to get to the metro station before they do. Luckily, you know some secret passageways and can perhaps take a shortcut.

You are given the map to EPFL as an n x m grid, with some obstacles. Esplanade is at the top left corner with index (0, 0), and the metro station is at the bottom right corner with index (n - 1, m - 1).

The Martians (and you) can move vertically or horizontally on the grid, and need 1 minute to go from one cell to another. The cells that are obstacles cannot be traversed.

You are also given k coordinates. These are the obstacles that you can walk (or run!) through but the Martians can't because they don't know about.

Can you get to the metro station before them?

Input

The first line will contain three space-separated integers n, m and k $(1 \leq n, m \leq 500;\ 1 \leq k \leq 10^5)$.

Each of the next n lines will have a string s_i of length m, representing row i in the EPFL map. Each character in s_i will be either a '.' representing a clear passage or a 'x' representing an obstacle.

Each of the next k lines will contain two space-separated integers r and c, where (r, c) represents an obstacle in the original graph that you can use as a secret passage $(0 \leq r \leq n-1;\ 0 \leq c \leq m-1)$.

All k (r, c) are obstacles in the original graph, and are pair-wise distinct.

The top left and bottom right corners of the EPFL map will not be obstacles.

Output

Print on a single line the word `YES` if you can reach the metro station strictly before the Martians do. Otherwise, print `NO`.

Examples

standard input	standard output

```
5 5 1
.....
xxxx.
.....
.xxxx
.....
3 4
```
	`YES`

```
5 5 1
..x..
...x.
x.x..
..x.x
.x.x.
3 4
```
	`NO`

Note

In the first example, the Martians need 16 minutes to reach the metro station. You, on the other hand, know the secret passage through (3, 4), which allows you to reach the metro station in 8 minutes. Even though you started 5 minutes after the Martians, you still arrive before them!

In the second example, neither you nor the Martians can reach the metro station. If only obstacle (0, 2) was also a secret passage, you would have been able to reach.

Problem C Salient Strings

Input file:	`standard input`
Output file:	`standard output`
Time limit:	30 seconds
Memory limit:	1024 megabytes

Many people are avid fans of the daily crossword in the paper, but not you. I mean, the format is pretty terrible, right? You only get to use English words, and any hack can look those up in a dictionary. Also, it takes forever to make just one puzzle. What a waste of time.

You've written a letter to the editor describing a new word game. It's really easy to make new puzzles because the only thing you give the solver is a permutation $P_{1..N}$ of the first N positive integers. It's then up to the solver to find any string that's salient for the given permutation.

A string is salient for the permutation $P_{1..N}$ if it consists of N uppercase letters ("A" ..."Z"), such that when its N non-empty suffixes are sorted in lexicographical order, the suffix starting at the i-th character is the P_i-th suffix in the sorted list. It's possible that a given permutation has no salient strings.

You need some example puzzles to include in your letter. You already have some permutations generated, so all you need is to supply an answer for each permutation (if possible).

Input

Input begins with an integer $T(1 \leq T \leq 2000)$, the number of different permutations you've generated. For each permutation, there is first a line containing the integer $N(1 \leq N \leq 1000)$. Then N lines follow, the i-th of which contains the integer $P_i(1 \leq P_i \leq N)$. It is guaranteed that each integer from 1 to N shows up exactly once in P.

Output

For the i-th permutation, print a line containing ``Case #i: '' followed by any salient string for that permutation (note that any valid string consisting of $N$ uppercase letters will be accepted), or ``-1'' if there are no such strings.

Example

standard input	standard output
3	Case #1: FACEBOOK
8	Case #2: FOXEN
5	Case #3: HUEHUEHUE
1	
3	
4	
2	
8	
7	
6	
5	
2	
4	
5	
1	
3	
9	
6	
9	
3	
5	
8	
2	
4	
7	
1	

Note

In the first case, if we sort the suffixes of FACEBOOK we get:

1. ACEBOOK
2. BOOK
3. CEBOOK
4. EBOOK
5. FACEBOOK
6. K
7. OK
8. OOK

If we read the indices of the suffixes back in order of decreasing length, we get 5 1 3 4 2 8 7 6, which is the given permutation. Therefore “FACEBOOK” is salient for this permutation, and is one possible accepted answer.

Problem D. Rain Over New York

Input file:	`standard input`
Output file:	`standard output`
Time limit:	30 seconds
Memory limit:	1024 megabytes

You're looking at a sort of clock which has a row of N lights. Each light is either on or off, and their states can be read as an N-digit binary number. The first light represents the most significant (leftmost) digit while the N-th light represents the least significant digit. A light that's on corresponds to a 1, and a light that's off corresponds to a 0.

You've just started looking at the clock, and you know that every second from now on, it will count upwards by 1, with its lights turning on or off to display the next number in binary. Once the clock displays $2^N - 1$ (with all N lights on), it will wrap around to display 0 (with all lights off) on the following second, and then continue counting up again.

However, 0 or more of the clock's lights may be permanently broken. You don't know which ones those are, but you know that they'll always appear to be off, even when they should be on.

Currently, the i-th light appears to be on if $L_i = 1$, and otherwise appears to be off (if $L_i = 0$). You have also received insider information that exactly K of the lights are currently supposed to be on. It's guaranteed that K is at least as large as the number of lights which appear to be on.

Assuming you stand around and look at this clock for a while, what's the maximum amount of time you might have to wait before you can be completely sure of what state every single light is currently supposed to be in? It's possible that you can be sure immediately, after 0 seconds. On the other hand, it's also possible that you might never be sure, no matter how long you wait.

Input

Input begins with an integer $T (1 \le T \le 5000)$, the number of different clocks you own. For each clock, there is first a line containing the two space-separated integers, $N (1 \le N \le 60)$ and $K (0 \le K \le N)$. Then there is a line containing N space-separated integers, the i-th of which is $L_i (0 \le L_i \le 1)$.

Output

For the i-th clock, print a line containing "`Case #i: `" followed by the maximum number of seconds which might go by before you know the true current state of each light, or -1 if you might never know.

Example

standard input

```
5
3 2
1 0 0
3 2
0 0 1
7 4
0 1 0 0 1 0 0
20 15
1 1 0 0 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1
20 19
0 1 1 0 1 0 0 0 1 1 0 0 0 1 1 0 1 0 1 0
```

standard output

```
Case #1: 2
Case #2: -1
Case #3: 19
Case #4: 217601
Case #5: 8193
```

Note

In the first case, exactly one of the two rightmost lights is initially supposed to be on, so the clock is either supposed to be showing 101 (5) or 110 (6). After one second, the clock will be supposed to show either 110 (6) or 111 (7). Supposing that the two rightmost lights are both broken, both of these values will still look like 100 to you, so you won't be able to tell which one is correct. After one more second, the clock will be supposed to show either 111 (7) or 000 (0). At that point, by observing the functioning leftmost light, you can determine whether all three lights are supposed to be on or off at that moment.

In the second case, you'll never be able to tell which of the two leftmost lights are supposed to be on if they're both broken.

Problem E Pick Your Team

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

This is it. The final battle between EPFL and Mars. The rules of the game are as follows.

Neither side wants to sacrifice their own people, so we will be picking two teams of Unil students to fight each other instead. You have been chosen to pick the team that will fight for EPFL's honour!

You are given a list containing the strength of each Unil student. You start by choosing one student to join your team, then the Martians will choose another student, and so on, until all n students are chosen.

If you had no extra information, clearly you'd pick the strongest Unil student in each turn. However, we managed to figure out the preference of the Martians. More specifically, we have a permutation P of the first n numbers, representing the indices of the Unil students, which the Martians prefer to pick in order.

Take a look at the example inputs to understand this further.

You want to pick the team that maximises the difference between your team's strength and theirs. What's the maximum difference?

Input

The first line of the input has of an even integer n ($2 \leq n \leq 100$), the number of Unil students.

The next line contains n space-separated integers s_i, the strength of each student ($1 \leq s_i \leq 10^7$).

The last line contains n space-separated integers between 1 and n, representing the permutation P.

Output

Print the maximum difference in strength between your team and the Martians' team.

Examples

standard input	standard output

`4`
`3 9 1 7`
`4 1 2 3`
	`12`

`10`
`1 1 2 3 4 5 6 6 8 10`
`9 8 7 6 5 4 10 1 2 3`
	`14`

Note

In the first example, there are four Unil students with strengths 3, 9, 1, 7.

The Martians prefer to pick them in this order: 4, 1, 2, 3.

This means that in their first turn, they'll pick student 4 (strength = 7) if that student hadn't been picked, otherwise they'll pick the next student on their list (student 1, strength = 3).

If you had used the simple strategy of picking the strongest available student each turn, you'd have ended up with a total strength of 9 + 3 = 12, and the Martians with 7 + 1 = 8, giving you a difference of 4.

Given this extra information, you can first pick student 4 (strength = 7), then student 2 (strength = 9) in your next turn. You'd have a difference of 9 + 7 - 3 - 1 = 12. In this case, this is the best strategy.

Problem F Central Element

Input file:	`standard input`
Output file:	`standard output`

This is an interactive problem.

There is a permutation P of numbers 1 through n, not known to you, $P = \langle P_1, P_2, ..., P_n \rangle$. You can ask the following type of questions: Given three distinct positions i, j and k, which of P_i, P_j and P_k is central? Element is central if it is neither minimal nor maximal.

For example, if the permutation is $\langle 2, 1, 4, 3 \rangle$, and you ask about positions 1, 2, and 3, you receive 2, because 2 is the central element of the set $\{P_1, P_2, P_3\} = \{2, 1, 4\}$. Note that you don't get the information at which position among 1, 2, and 3 it is located.

Your task is to find the permutation P. Actually, for each permutation P there is a set $S(P)$ of permutations that cannot be distinguished from P using the allowed questions. You must find any permutation from this set.

Interaction protocol

First, your program must read from the standard input one line with the integer n, the size of the permutation.

The program must write to the standard output one line with three positions that you ask a question about and wait for a line in the standard input with a response, then write next question and read next response, and so on until you know the permutation P up to $S(P)$.

Once you know the answer, output one line with the word "`OK`" and the permutation P.

Input

The first line of the standard input contains n, the size of the permutation ($3 \le n \le 200$).

Each of the next lines of the standard input contains response to your question — the number that is central among the numbers at the asked positions.

Output

When you're asking questions, each line of the standard output should contain three different integers from the range of 1 to n, space-separated. You can ask at most 2 000 questions.

When you're stating the answer, the line of the standard output should contain the word "`OK`", and the numbers $P_1, P_2, \ldots, P_n$, all space-separated. After printing this line your program must exit.

You must flush standard output after printing each line.

Sample input and output

standard input	standard output
4	1 2 3
2	2 3 4
3	1 2 4
2	1 3 4
3	OK 2 1 4 3