icpc International Collegiate Programming Contest

The 2025 ICPC Europe Championship

Problem Set

icpc.foundation

icpc Europe Championship

U.PORTO

host university

JETBRAINS

HUAWEI

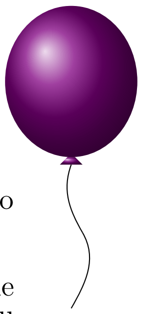
Jane Street

Jump TRADING

pinely

WINCENT

A Condorcet Elections



It is a municipality election year. Even though the leader of the country has not changed for two decades, the elections are always transparent and fair.

There are n political candidates, numbered from 1 to n, contesting the right to govern. The elections happen using a variation of the *Ranked Voting System*. In their ballot, each voter will rank all n candidates from most preferable to least preferable. That is, each vote is a permutation of $\{1, 2, \ldots, n\}$, where the first element of the permutation corresponds to the most preferable candidate.

We say that candidate a defeats candidate b if in more than half of the votes candidate a is more preferable than candidate b.

As the election is fair and transparent, the state television has already decreed a list of m facts—the i-th fact being "candidate a_i has defeated candidate b_i"—all before the actual election!

You are in charge of the election commission and tallying up the votes. You need to present a list of votes that produces the outcome advertised on television, or to determine that it is not possible. However, you are strongly encouraged to find a solution, or you might upset higher-ups.

Input

The first line contains integers n and m ($2 \le n \le 50$, $1 \le m \le \frac{n(n-1)}{2}$) — the number of parties and the number of pairs with known election outcomes.

The i-th of the following m lines contains two integers a_i and b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$) — candidate a_i defeats candidate b_i.

Each unordered pair $\{a_i, b_i\}$ is given at most once.

Output

Print `YES` if there is a list of votes matching the facts advertised on television. Otherwise, print `NO`.

If there is a valid list of votes, print one such list in the following lines.

Print the number k of votes cast ($1 \le k \le 50\,000$). It can be shown that if there is a valid list of votes, there is one with at most $50\,000$ votes.

Then print k lines. The i-th of these lines consists of a permutation of $\{1, 2, \ldots, n\}$ describing the i-th vote. The first number in the permutation is the most preferable candidate and the last one is the least preferable candidate.

For $1 \le i \le m$, a_i shall appear earlier than b_i in more than $k/2$ of the k permutations. For pairs of candidates $\{a, b\}$ not appearing in the election requirements list, the outcome can be arbitrary, including neither of a and b defeating the other.

Samples

Sample input 1	Sample output 1
2 1 1 2	YES 1 1 2

Sample input 2	Sample output 2

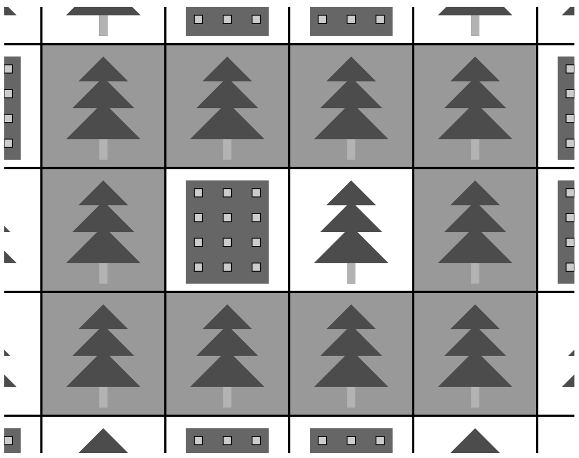
3 3
1 2
2 3
3 1
	YES
	3
	1 2 3
	2 3 1
	3 1 2

Explanation of sample 2.
Observe that candidate 1 defeats candidate 2 because it goes earlier in two out of three voters' permutations, which is more than half of all votes. Similarly, candidate 2 defeats candidate 3, and candidate 3 defeats candidate 1.

B Urban Planning

You are responsible for planning a new city! The city will be represented by a rectangular grid, where each cell is either a park or a built-up area.

The residents will naturally want to go for walks in the city parks. In particular, a *rectangular walk* is a rectangle consisting of the grid cells, which is at least 2 cells long both horizontally and vertically, such that all cells on the boundary of the rectangle are parks. Note that the cells inside the rectangle can be arbitrary.


An example rectangular walk (cells with dark background).

Your favourite number is k. To leave a long-lasting signature, you want to design the city in such a way that it has exactly k rectangular walks.

Input

The input contains a single integer k ($0 \le k \le 4 \cdot 10^{12}$).

Output

On the first line, print two integers h and w ($1 \le h, w \le 2025$), the height and width of the grid. On the next h lines, print a string with w characters each, with each character being #, denoting a park, or ., denoting a built-up area.

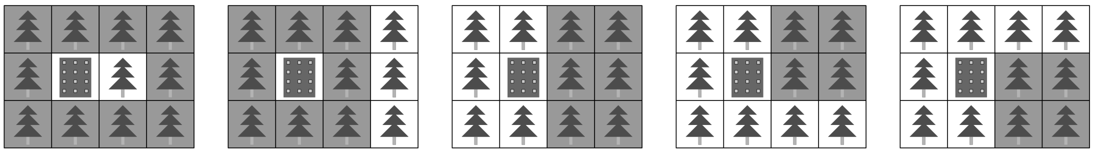
It is guaranteed that for any value of k within the given limits, there exists a solution with height and width within the given limits. Any city within the given limits and with exactly k rectangular walks will be accepted.

Samples

Sample input 1	Sample output 1
5	3 4 #### #.## ####

Explanation of sample 1.

Here are the five possible rectangular walks:



C Ads

You have n videos on your watchlist on the popular platform YooCube. The i-th video lasts d_i minutes.

YooCube has recently increased the frequency of their ads. Ads are shown only between videos. After finishing a video, an ad is shown if either of these two conditions is true:

- three videos have been watched since the last ad;
- at least k minutes have passed since the end of the last ad.

You want to watch the n videos in your watchlist. Given that you have just watched an ad, and that you can choose the order of the n videos, what is the minimum number of ads that you are forced to watch? You can start a new video immediately after the previous video or ad ends, and you don't have to watch any ad after you finish.

Input

Each test contains multiple test cases. The first line contains an integer t $(1 \le t \le 100\,000)$ — the number of test cases. The descriptions of the t test cases follow.

The first line of each test case contains two integers n and k $(1 \le n \le 100\,000, 1 \le k \le 30\,000)$ — the number of videos in your watchlist and the parameter that determines when ads are shown.

The second line contains n integers $d_1, d_2, \ldots, d_n (1 \le d_i \le 10\,000)$ — the lengths of the videos.

The sum of the values of n over all test cases does not exceed 10^6.

Output

For each test case, print the minimum number of ads that you need to watch.

Samples

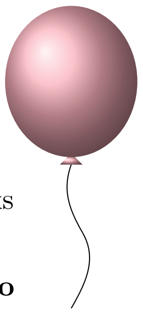
Sample input 1	Sample output 1

5
8 25
4 5 18 3 17 17 18 14
7 21
20 14 1 4 20 8 4
8 1
20 5 9 4 14 12 2 20
8 37
2 13 13 11 12 19 16 18
4 38
15 3 14 7
	2
	2
	7
	2
	1

Explanation of sample 1.

In the **first test case**, a possible viewing order is $4, 1, 8, 2, 5, 6, 7, 3$ (the corresponding lengths being $3, 4, 14, 5, 17, 17, 18, 18$). With this order, you will have to watch an ad after the first three videos and then another after the second three videos. Note that you don't have to watch an ad after you finish watching all your videos.

D Morse Code



Morse code is a classical way to communicate over long distances, but there are some drawbacks that increase the transmission time of long messages.

In Morse code, each character in the alphabet is assigned a sequence of dots and dashes such that **no sequence is a prefix of another**. To transmit a string of characters, the sequences corresponding to each character are sent in order. **A dash takes twice as long to transmit as a dot.**

Your alphabet has n characters, where the i-th character appears with frequency f_i in your language. Your task is to design a Morse code encoding scheme, assigning a sequence of dots and dashes to each character, that minimizes the expected transmission time for a single character. In other words, you want to minimize $f_1 t_1 + f_2 t_2 + \cdots + f_n t_n$, where t_i is the time required to transmit the sequence of dots and dashes assigned to the i-th character.

Input

The first line contains an integer n ($2 \leq n \leq 200$) — the number of characters in the alphabet.

The second line contains n real numbers $f_1, f_2, \ldots, f_n$ ($0 < f_i < 1$) — f_i is the frequency of the i-th character. All values $f_1, f_2, \ldots, f_n$ are given with exactly four digits after the decimal point. The sum of all frequencies is exactly 1.

Output

Print n lines, each containing one string consisting of dots `.` and dashes `-`. The i-th line corresponds to the sequence of dots and dashes that you assign to the i-th character.

If there are multiple valid assignments with the minimum possible expected transmission time, any of them is considered correct.

Samples

Sample input 1	Sample output 1

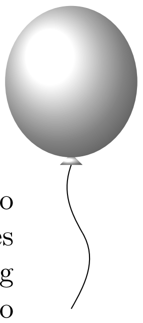
`3`
`0.3000 0.6000 0.1000`
	`-.`
	`.`
	`--`

Explanation of sample 1.
The alphabet contains three letters, say a, b, and c, with respective frequencies 0.3, 0.6, and 0.1. In the optimal assignment, we assign a to '`-.`', b to '`.`', and c to '`--`'. This gives an expected transmission time of $0.3 \cdot 3 + 0.6 \cdot 1 + 0.1 \cdot 4 = 1.9$ time units per character, which is optimal.

For comparison, the assignment $a \to$ '`..`', $b \to$ '`-`', $c \to$ '`.-`' has an expected transmission time of $0.3 \cdot 2 + 0.6 \cdot 2 + 0.1 \cdot 3 = 2.1$. The assignment $a \to$ '`-`', $b \to$ '`.`', $c \to$ '`..`' has a lower expected transmission time, but is invalid since '`.`' is a prefix of '`..`'.

Sample input 2	Sample output 2
3 0.3000 0.4500 0.2500	.. - .-

$\boxed{\text{E}}$ Porto Vs. Benfica



FC Porto and SL Benfica are the two largest football teams in Portugal. Naturally, when the two play each other, a lot of people travel from all over the country to watch the game. This includes the Benfica supporters' club, which is going to travel from Lisbon to Porto to watch the upcoming game. To avoid tensions between them and the Porto supporters' club, the national police want to delay their arrival to Porto as much as they can.

The road network in Portugal can be modelled as a simple, undirected, unweighted, connected graph with n vertices and m edges, where vertices represent towns and edges represent roads. Vertex 1 corresponds to Lisbon, i.e., the starting vertex of the supporters' club, and vertex n is Porto, i.e., the destination vertex of the supporters' club. The supporters' club wants to minimize the number of roads they take to reach Porto.

The police are following the supporters' club carefully, and so they always know where they are. To delay their arrival, at any point the police can pick exactly one road and block it, as long as the supporters' club isn't currently traversing it. They can do this exactly once, and once they do that, the road is blocked forever. Once the police block a road, the supporters' club immediately learns that that road is blocked, and they can change their route however they prefer. Furthermore, the supporters' club knows that the police are planning on blocking some road and can plan their route accordingly.

Assuming that both the supporters' club and the police always make optimal choices, determine the minimum number of roads the supporters' club needs to traverse to go from Lisbon to Porto. If the police can block the supporters' club from ever reaching Porto, then output -1.

Input

The first line contains two integers n and m ($2 \leq n \leq 200\,000$, $n-1 \leq m \leq \min\{n(n-1)/2, 200\,000\}$) — the number of towns and the number of roads in the road network of Portugal.

Each of the next m lines contains two integers s_i and t_i ($1 \leq s_i, t_i \leq n$) — the two towns connected by the i-th road.

It is guaranteed that the road network is connected, each road connects two distinct towns, and that there are no repeated roads.

Output

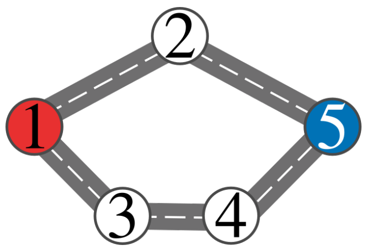
Print the minimum number of roads the supporters' club needs to traverse to travel from Lisbon to Porto.

Samples

Sample input 1	Sample output 1

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

Explanation of sample 1.
The road network is represented by the following picture:

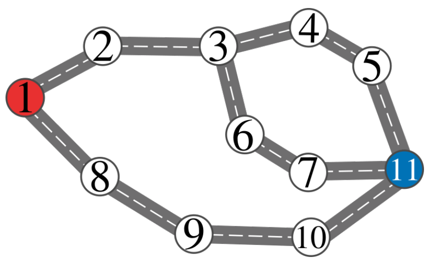


Note that the optimal strategy for the police is to wait until the supporters' club is on a vertex adjacent to the destination (i.e., vertex 5) and then block the edge that connects that vertex to the destination. The optimal strategy for the supporters' club is to follow the upper path (by going from 1 to 2) and then after seeing the edge between 2 and 5 get blocked, they will go around by following 2 back to 1 and then following the lower path (1 to 3 to 4 to 5). The number of traversed roads in this case is 5.

Sample input 2	Sample output 2

11 12
1 2
2 3
3 4
4 5
5 11
3 6
6 7
7 11
1 8
8 9
9 10
10 11
	9

Explanation of sample 2.
The road network is represented by the following picture:

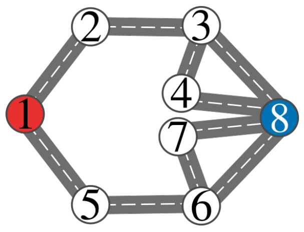


There are multiple strategies here, but one can verify that the optimal one for both the police and supporters' club is to follow the upper path ($1 \to 2 \to 3 \to 4 \to 5$), then the police block the edge 5 to 11, and so the supporters' club go around by following $5 \to 4 \to 3 \to 6 \to 7 \to 11$. The total number of traversed roads of this strategy is 9.

Sample input 3	Sample output 3

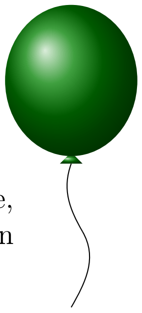
```
8 10
1 2
2 3
3 4
3 8
4 8
1 5
5 6
6 7
6 8
7 8
```
	`5`

Explanation of sample 3.
The road network is represented by the following picture:



The optimal strategy for the police is to block edge 2 to 3 if the supporters' club reaches vertex 2, or block edge 5 to 6 if the supporters' club reaches vertex 5. The optimal path in this case is $1 \to 2 \to 1 \to 5 \to 6 \to 8$. The total number of traversed roads in this case is 5.

F Mascot Naming



When organizing a big event, organizers often handle side tasks outside their expertise. For example, the chief judge of EUC 2025 must find a name for the event's official mascot while satisfying certain constraints:

- The name must include specific words as subsequences*, such as the event name and location. You are given the list s_1, s_2, ..., s_n of the n required words.
- The name must not contain as a subsequence* the name t of last year's mascot.

Please help the chief judge find a valid mascot name or determine that none exists.

* A string x is a *subsequence* of a string y if x can be obtained from y by erasing some characters (at any positions) while keeping the remaining characters in the same order. For example, `abc` is a subsequence of `axbycz` but not of `acbxyz`.

Input

The first line contains an integer n ($1 \le n \le 200\,000$) — the number of words that shall appear as subsequences.

The i-th of the following n lines contains the string s_i ($1 \le |s_i| \le 200\,000$, s_i consists of lowercase English letters) — the i-th word in the list of words that shall appear as subsequences. The total length of these n words is at most 200 000, i.e., $|s_1| + |s_2| + \cdots + |s_n| \le 200\,000$.

The last line contains the string t ($1 \le |t| \le 200\,000$, t consists of lowercase English letters) — the name of last year's mascot.

Output

Print `YES` if there is a valid name for the mascot. Otherwise, print `NO`.

If there is a valid name, on the next line print a valid name. The string you print must have length at most 1 000 000 and must consist of lowercase English letters. One can prove that if a valid name for the mascot exists, then there is one satisfying these additional constraints.

If there are multiple solutions, print any of them.

Samples

Sample input 1	Sample output 1
2 porto euc prague	YES poretuco

Explanation of sample 1.

The words that must appear as subsequences are `porto` and `euc`, while the word that must not appear as a subsequence is `prague`. There are many valid names for the mascot, e.g., `poretuco` or `xxxpppeortoucyyyy`.

If `poretuco` is chosen as the name of the mascot, observe that `porto` and `euc` are subsequences (highlighted in `POReTucO` and `porEtUCo`, respectively), while `prague` is not, as desired.

The string `poretuc` is not a valid mascot name because it does not contain the word `porto` as a subsequence. Also the string `poretucoague` is not a valid mascot name because it contains `prague` as a subsequence.

Sample input 2	Sample output 2

`6`
`credit`
`debit`
`money`
`rich`
`bank`
`capitalism`
`trap`
	`YES`
	`moncrdebdziteychankpitalism`

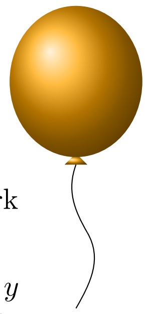
Sample input 3	Sample output 3

`2`
`axiom`
`choice`
`io`
	`NO`

Sample input 4	Sample output 4

`4`
`aaa`
`aab`
`abb`
`bbb`
`ba`
	`YES`
	`aaabbb`

G A Very Long Hike



You are planning a hike in the Peneda-Gerês National Park in the north of Portugal. The park takes its name from two of its highest peaks: Peneda (1340 m) and Gerês (1545 m).

For this problem, the park is modelled as an infinite plane, where each position (x, y), with x, y being integers, has a specific altitude. The altitudes are defined by an $n \times n$ matrix h, which repeats periodically across the plane. Specifically, for any integers a, b and $0 \le x, y < n$, the altitude at $(x + an, y + bn)$ is $h[x][y]$.

When you are at position (x, y), you can move to any of the four adjacent positions: $(x, y + 1)$, $(x + 1, y)$, $(x, y - 1)$, or $(x - 1, y)$. The time required to move between two adjacent positions is $1 + |\text{alt}_1 - \text{alt}_2|$, where alt_1 and alt_2 are the altitudes of the current and destination positions, respectively.

Initially, your position is $(0, 0)$. Compute the number of distinct positions you can reach within 10^{20} seconds. Your answer will be considered correct if its relative error is less than 10^{-6}.

Input

The first line contains an integer n $(2 \le n \le 20)$—the size of the matrix describing the altitudes.

The following n lines contain n integers each. The $(j+1)$-th number on the $(i+1)$-th of these lines is $h[i][j]$ $(0 \le h[i][j] \le 1545)$—the altitude of the position (i, j).

Output

Print the number of distinct positions you can reach within 10^{20} seconds. Your answer will be considered correct if its relative error is less than 10^{-6}.

Samples

Sample input 1	Sample output 1

2
3 3
3 3
	2e+40

Explanation of sample 1.
Every position of the Peneda-Gerês National Park has an altitude of 3. Therefore, the time required to move between two adjacent positions is always equal to 1 second.

In this case, one can show that a position (x, y) is reachable within 10^{20} seconds if and only if $|x| + |y| \le 10^{20}$. One can compute that there exist 20 000 000 000 000 000 000 200 000 000 000 000 000 001 reachable positions and this number is approximated by $2 \cdot 10^{40}$ with a relative error smaller than 10^{-6}. The sample output shows $2 \cdot 10^{40}$ as correct answer, but also the exact number of reachable positions would be a correct answer.

Sample input 2	Sample output 2

`3`
`0 0 0`
`0 1545 0`
`0 0 0`
	`2e+40`

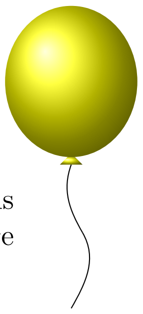
Explanation of sample 2.
Every position (x, y) of the Peneda-Gerês National Park with $x - 1$ and $y - 1$ divisible by 3 has an altitude of 1545, while all the other positions have an altitude of 0. For example, the time required to move between $(4, 10)$ and $(4, 9)$ is 1546, while the time required to move between $(3, 2)$ and $(4, 2)$ is 1.

The positions reachable in 2 seconds are all positions (x, y) with $|x|+|y| \leq 2$ apart from $(1, 1)$ (which is on the peak). One can compute that there exist 19 999 999 999 999 999 931 533 333 333 333 333 863 441 reachable positions in 10^{20} seconds and this number is approximated by $2 \cdot 10^{40}$ with a relative error smaller than 10^{-6}. The sample output shows $2 \cdot 10^{40}$ as correct answer, but also the exact number of reachable positions would be a correct answer.

Sample input 3	Sample output 3

`4`
`0 1 2 3`
`5 6 7 4`
`10 11 8 9`
`15 12 13 14`
	`1.524886878e+39`

H Statues



The mayor of a city wants to place n statues at intersections around the city. The intersections in the city are at all points (x, y) with integer coordinates. Distances between intersections are measured using Manhattan distance, defined as follows:

$$\text{distance}((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|.$$

The city council has provided the following requirements for the placement of the statues:

- The first statue is placed at $(0, 0)$;
- The n-th statue is placed at (a, b);
- For $i = 1, \ldots, n-1$, the distance between the i-th statue and the $(i+1)$-th statue is d_i.

It is allowed to place multiple statues at the same intersection.

Help the mayor find a valid arrangement of the n statues, or determine that it does not exist.

Input

The first line contains an integer n $(3 \le n \le 50)$ — the number of statues.

The second line contains two integers a and b $(0 \le a, b \le 10^9)$ — the coordinates of the intersection where the n-th statue must be placed.

The third line contains $n-1$ integers $d_1, \ldots, d_{n-1}$ $(0 \le d_i \le 10^9)$ — the distance between the i-th statue and the $(i+1)$-th statue.

Output

Print `YES` if there is a valid arrangement of the n statues. Otherwise, print `NO`.

If there is a valid arrangement, print a valid arrangement in the following n lines. The i-th of these lines must contain two integers x_i and y_i — the coordinates of the intersection where the i-th statue is placed. You can print any valid arrangement if multiple exist.

Samples

Sample input 1	Sample output 1

3
5 8
9 0
	NO

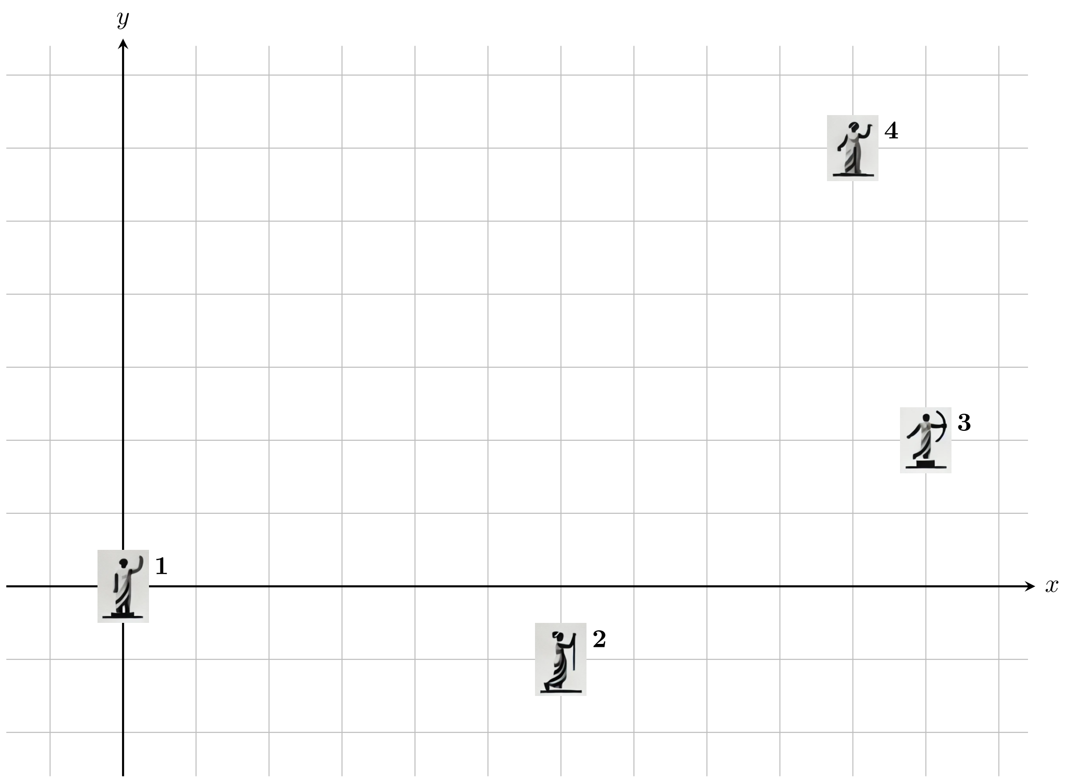
Explanation of sample 1.
There is no valid arrangement of the 3 statues.

Sample input 2	Sample output 2

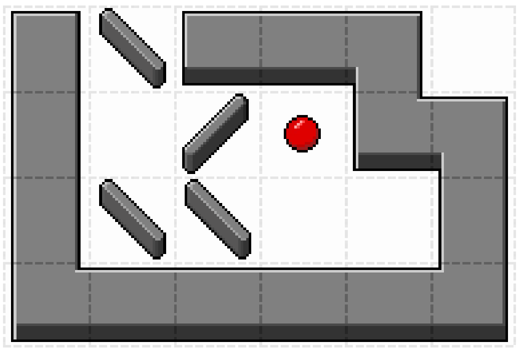
4
10 6
7 8 5
	YES
	0 0
	6 -1
	11 2
	10 6

Explanation of sample 2.

The sample output is shown in the following picture. Note that this is not the only valid arrangement of the 4 statues.



I Pinball



You are playing a pinball-like game on a $h \times w$ grid.

The game begins with a small ball located at the center of a specific cell marked as `S`. Each cell of the grid is either:

- A block-type wall (`#`) that prevents the ball from entering the cell, reflecting it instead.
- A thin oblique wall, either left-leaning (`\`) or right-leaning (`/`), which reflects the ball according to its orientation.
- A free cell (`.`) where the ball can move freely.

The goal is to make the ball escape the grid.

At the start, you can nudge the ball in one of four directions: up (`U`), down (`D`), left (`L`), or right (`R`). The ball traverses a free cell in one second, it enters and exits a cell containing a thin oblique wall in one second, and it bounces off a block-type wall in no time (the block-type wall occupies all of its cell).

Collisions between the ball and all walls, both block-type and oblique, are perfectly elastic, causing the ball to reflect upon contact.

For example, the ball takes two seconds to enter a free cell, traverse it, bounce off an adjacent block-type wall, and traverse back the free cell until it exits.

As the ball moves, you may destroy oblique walls at any time, permanently converting them into free cells. You may destroy multiple oblique walls throughout the game, at any given time.

Determine whether it is possible for the ball to escape, and if so, find the **minimum** number of oblique walls that need to be destroyed, along with the precise time each chosen wall should be destroyed.

Input

The first line contains two integers h and w ($1 \leq h, w \leq 1000$) — the size of the grid.

The next h lines describe the grid at the beginning of the game.

The i-th of these lines contains w characters, describing the cells on the i-th row. A dot (`.`) denotes a free cell, a hash sign (`#`) denotes a block-type wall, a (`\`) or (`/`) denotes a thin oblique wall, and an `S` denotes the starting position of the ball (the starting position is a free cell).

It is guaranteed that all the $h \cdot w$ characters describing the grid belong to the set {`.`, `#`, `\`, `/`, `S`} and the character `S` appears exactly once.

Output

Print `YES` if it is possible to make the ball escape the grid. Otherwise, print `NO`.

If it is possible, print the following extra information.

On the second line, print a single character $d \in \{\texttt{U}, \texttt{D}, \texttt{L}, \texttt{R}\}$ — the direction of the starting nudge of the ball.

On the third line, print k — the minimum number of oblique walls to be destroyed.

On each of the following k lines, print three integers t_i, r_i, and c_i — the oblique wall in the cell on the r_i-th row from the top and on the c_i-th column from the left is destroyed t_i seconds after the ball starts moving. Note that the corresponding wall is destroyed **just before** t_i seconds have elapsed, essentially meaning that the corresponding cell acts as a free cell if the ball would have hit that wall exactly t_i seconds after the start.

The operations must be printed in chronological order (i.e., $t_i \le t_{i+1}$ for all $1 \le i \le k-1$). The same wall cannot be destroyed multiple times (i.e., $(r_i, c_i) \ne (r_j, c_j)$ if $i \ne j$). Initially, there must be an oblique wall at the cell identified by (r_i, c_i) for all $1 \le i \le k$.

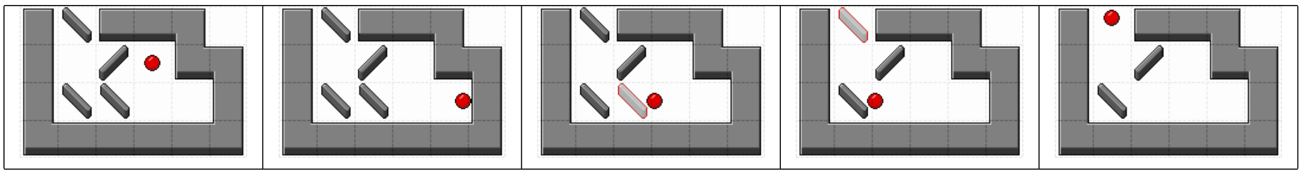
All t_i must satisfy $0 \le t_i \le 10^7$. It can be proved that, if there exists a solution, there exists a solution where no t_i exceeds 10^7.

Samples

Sample input 1	Sample output 1

```
4 6
#\###.
#./S##
#\\..#
######
```
	```
	YES
	L
	2
	7 3 3
	8 1 2
	```

Explanation of sample 1.
The minimum number of walls to be destroyed is 2. We describe the relevant moments of the solution given as sample output.



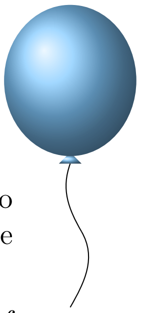
- At time $t = 0$, the ball is in its initial position and gets nudged towards the **left** direction.
- At time $t = 4.5$, the ball hits a block-type wall and reflects off of it.
- Right before $t = 7$, the wall at position $(3, 3)$ is destroyed.
- Right before $t = 8$, the wall at position $(1, 2)$ is destroyed.
- At time $t = 10.5$, the ball finally escapes the grid.

Sample input 2	Sample output 2

```
3 3
###
.S.
###
```
	```
	YES
	R
	0
	```

Explanation of sample 2.
You can just nudge the ball towards the left or right direction, and the ball will eventually escape the grid.

J The Ultimate Wine Tasting Event



Rumors of the excellence of Gabriella's wine tasting events have toured the world and made it to the headlines of prestigious wine magazines. Now, she has been asked to organize an event at the EUC 2025!

This time she selected $2n$ bottles of wine, of which exactly n are of white wine, and exactly n of red wine. She arranged them in a line as usual, in a predetermined order described by a string s of length $2n$: for $1 \le i \le 2n$, the i-th bottle from the left is white wine if $s_i = \texttt{W}$ and red wine if $s_i = \texttt{R}$.

To spice things up for the attendees (which include EUC contestants), Gabriella came up with the following wine-themed problem:

Consider a way of dividing the $2n$ bottles into two disjoint subsets, each containing n bottles. Then, for every $1 \le i \le n$, swap the i-th bottle in the first subset (from the left) and the i-th bottle of the second subset (also from the left). Is it possible to choose the subsets so that, after this operation is done exactly once, the white wines occupy the first n positions?

Input

The first line contains an integer t ($1 \le t \le 500$) — the number of test cases. The descriptions of the t test cases follow.

The first line of each test case contains an integer n ($1 \le n \le 100$) — where $2n$ is the total number of bottles.

The second line of each test case contains a string s of length $2n$, describing the bottle arrangement — the i-th character of s ($1 \le i \le 2n$) is `W` for a white wine and `R` for a red wine.

It is guaranteed that s contains exactly n `W`'s and n `R`'s.

Output

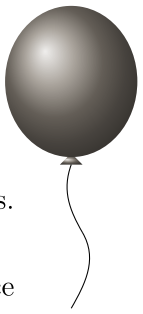
For each test case, print `YES` if it is possible to divide the bottles as explained in the statement. Otherwise, print `NO`.

Samples

Sample input 1	Sample output 1
3 4 WRRWWWRR 1 WR 20 WWWWRRWRRRRRWRRWRWRRWRRWWWWWWWRWWRWWRRRR	YES NO YES

Explanation of sample 1.
In the **first test case**, we can make one subset out of the bottles at positions 1, 2, 3 and 7 (which are, respectively: white, red, red, red) and the other subset out of the bottles at positions 4, 5, 6, 8 (which are, respectively: white, white, white, red). We will then swap pairs $(1, 4)$, $(2, 5)$, $(3, 6)$ and $(7, 8)$, after which the white wines will occupy the first 4 positions, and the red wines the last 4 positions.

In the **second test case**, there is only one possible way to form the subsets: one with the first bottle, and one with the second bottle. After swapping, the resulting arrangement is `RW`, therefore there is no solution.



K Amusement Park Rides

Ivan, Dmitrii, and Pjotr are celebrating Ivan's birthday at an amusement park with n attractions. The i-th attraction operates at minutes $a_i, 2a_i, 3a_i, \ldots$ (i.e., every a_i minutes).

Each minute, the friends can either ride exactly one available attraction **together** or wait. Since the rides are very short, they can board another attraction the next minute. They may ride the attractions in any order.

They want to experience each ride exactly once before heading off to enjoy the birthday cake. What is the earliest time by which they can finish all n attractions?

Input

Each test contains multiple test cases. The first line contains an integer t ($1 \le t \le 2000$) — the number of test cases. The descriptions of the t test cases follow.

The first line contains an integer n ($1 \le n \le 2000$) — the number of attractions.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the values determining when the various attractions operate.

It is guaranteed that the sum of n over all test cases does not exceed 2000.

Output

For each test case, print the earliest time the three friends can finish all n attractions.

Samples

Sample input 1	Sample output 1

3
4
1 2 3 4
4
1 1 1 1
6
1 2 1 2 2 2
	4
	4
	8

Explanation of sample 1.
In the **first test case**, the three friends can ride the i-th attraction at the i-th minute. Since there are 4 attractions, they need 4 minutes to ride them all.

In the **third test case**, the three friends can ride the attractions in order at minutes 1, 2, 3, 4, 6, 8 respectively. Therefore, they can ride all attractions in 8 minutes. One can show that it is impossible to finish all the attractions earlier.