Problem A
Amazing Adventures

The amazing game Pokenom Go has just been released. Pokenom trainers can now travel the world, capture Pokenom in the wild and battle each other!

Bash and Cee are two students who have recently dropped out of university to pursue their childhood dream of becoming Pokenom trainers. Today their amazing adventures begins!

The world can be considered as a grid with N rows and M columns. Rows are numbered from 1 to N from bottom to top, and columns are numbered from 1 to M from left to right. The cell at r-th row and c-th column is denoted as (r, c).

Bash's house is located at cell (r_B, c_B). Today he will go to the Pokenom Gym located at cell (r_G, c_G) where he can battle other Pokenom trainers. On the way to the Pokenom Gym, Bash also wants to visit Cee's house, located at cell (r_C, c_C).

Bash's university is located at cell (r_U, c_U). Bash does not want to pass the university on the path to the Pokenom Gym, as that will trigger Bash's bad memories.

In each step, Bash can only go in either 4 directions (up, down, left and right) to a cell which shares exactly one common edge with the current cells. Hence, a path can be uniquely defined by a string consisting of 4 characters 'U', 'D', 'L', 'R' — representing the directions up, down, left and right, respectively.

A simple path is a path where no cells (including the starting and ending cells) are visited more than once.

Bash wants to find a simple path from cell (r_B, c_B) to cell (r_G, c_G), which passes through cell (r_C, c_C) but not cell (r_U, c_U). Bash would like to use as few steps as possible. Please help Bash!

Input

The input contains multiple test cases. Each test case is described by 6 lines:

- The first line contains exactly 2 integers N and M, separated by a single space. $(1 \leq M, N \leq 100)$.
- The second line contains exactly 2 integers r_B and c_B, separated by a single space.
- The third line contains exactly 2 integers r_C and c_C, separated by a single space.
- The fourth line contains exactly 2 integers r_G and c_G, separated by a single space.
- The fifth line contains exactly 2 integers r_U and c_U, separated by a single space.
- The sixth line is a blank line.
- $1 \leq r_B, r_C, r_G, r_U \leq N, 1 \leq c_B, c_C, c_G, c_U \leq M$. The 4 cells (r_B, c_B), (r_C, c_C), (r_G, c_G) and (r_U, c_U) are pairwise different.

The input is terminated by a single line containing two zeros.

The sum of $M \cdot N$ over all test cases does not exceed 10^5.

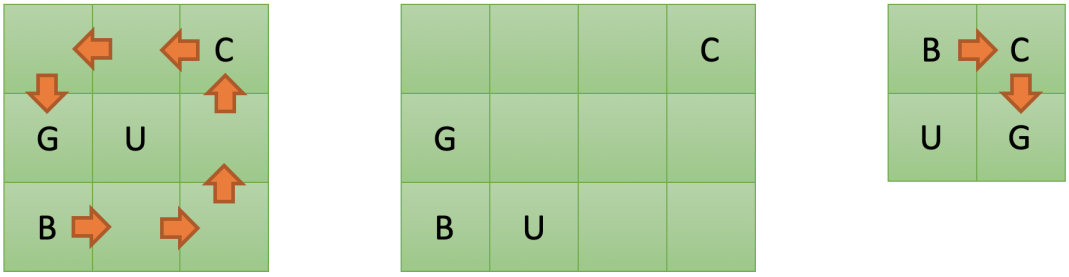
Output

For each test case, if it is impossible to find a path satisfying all the given constraints, print exactly one line with the word 'NO'. Otherwise, print 2 lines:

- The first line contains the word 'YES'.
- The second line contains a string presenting Bash's shortest simple path.

If there are multiple shortest simple paths, you can output any of them.

Sample Illustration



Sample Input 1	Sample Output 1
3 3 1 1 3 3 2 1 2 2 3 4 1 1 3 4 2 1 1 2 2 2 2 1 2 2 1 2 1 1 0 0	YES RRUULLD NO YES RD

Problem B
Bipartite Battle

A Bipartite Graph is an undirected graph whose vertices can be partitioned into 2 sets such that, for each edge (u, v), u and v belong to different sets.

Socket has challenged Bash to a Bipartite Battle. In the Bipartite Battle, Bash and Socket play a game with Bipartite Graphs.

The Bipartite Battle happens as follows:

- Socket draws N bipartite graphs. The i-th graph has 2 sets of vertices, one with a_i vertices, and the other with b_i vertices.
- Then the 2 players, Bash and Socket alternatively take turns. In each turn, a player must choose exactly one non-empty graph, then delete exactly one edge or exactly one vertex of the chosen graph. If the player deletes one vertex, all edges adjacent to it are also deleted.
- The player who cannot move loses. Note that this can only happen when all edges and all vertices of all graphs have been deleted.
- Bash plays first.

Of course, Socket does not want to play fair. Socket plans to draw bipartite graphs such that he always wins.

How many ways can Socket draw N bipartite graphs, so that he always wins, assuming both players play optimally?

Notes

For the i-th bipartite graph, let's number the vertices in first set from 1 to a_i, and the vertices in second set from 1 to b_i.

An edge connecting vertex u in first set and vertex v in second set is denoted as (u, v).

Two drawings of bipartite graphs are considered different iff there exists an index j and a pair of integers (u, v), such that:

- (u, v) is an edge of j-th graph in one drawing.
- (u, v) is NOT an edge of j-th graph in the other drawing.

Input

The first line of input contains the integer N $(1 \le N \le 10^5)$.

N lines follow, each line contains exactly 2 space-separated integers a_i and b_i $(1 \le a_i, b_i \le 10^9)$.

Output

Print exactly one integer — the number of ways Socket can draw N bipartite graphs, modulo $10^9 + 7$.

Sample Input 1	Sample Output 1
1 1 1	1

Sample Input 2	Sample Output 2
1 1 2	0

Problem C
Conquest Campaign

Since the beginning of 30*-th century, the Country of Circles has become the strongest country in the world. To expand its territory to the west, they plan to invade the Country of Rectangles.*

The territory of the Country of Rectangles is represented by a $R \times C$ table, where rows are numbered from 1 to R, and columns are numbered from 1 to C. The cell at the i-th row and j-th column is denoted as (i, j).

The Department of Defense of the Country of Circles plans to use its elite army of paratroopers to attack the Country of Rectangles. By sending spies to their opponent, they know that in the Country of Rectangles, N cells $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ are very weakly protected and can easily be dominated. Hence, they come up with the following plan:

- On the first day, they plan to send a battalion of paratroopers to occupy each of these N cells.
- On each of the following days, they plan to send reinforcement to occupy all cells which share a common edge with at least one previously occupied cell.

We assume that the Country of Circle's force is strong enough that they can occupy any cell that they want.

The commander wants to know how many days it would take to conquer the whole country.

Input

- The first line contains three integers R, C and N $(1 \le R, C \le 100,\ 1 \le N \le 10,000)$ — the number of rows, the number of columns of the Country of Rectangles' territory and the number of weakly protected cells, respectively. It is not guaranteed that the cells are unique.
- The i-th of the remaining N lines contains two integer x_i and y_i $(1 \le x_i \le R,\ 1 \le y_i \le C)$ — one cell where paratroopers are sent during the first day of the campaign.

Output

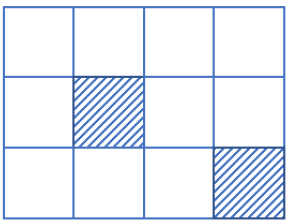
Print exactly one number — The number of days needed for the Country of Circles to completely conquer the Country of Rectangles.

Explanation for the first example

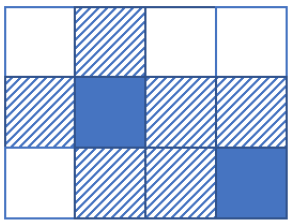
The figure below shows how the plan is operated for each day, where:

- Unoccupied cells are in white.
- Cells occupied on this day are filled with stripes.

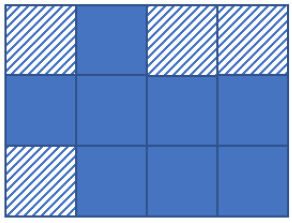
- Cells occupied on previous days are in solid color.



Day 1



Day 2



Day 3

Sample Input 1

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

Sample Output 1

```
3
```

Sample Input 2

```
2 3 6
1 1
1 2
1 3
2 1
2 2
2 3
```

Sample Output 2

```
1
```

Problem D
Divide Doughnut

This is an interactive problem.

Khue and Hanh are obsessed with doughnuts although they are very much overweight. Over the last week, they worked very hard setting up the problems for ICPC Hanoi. To prepare for a long night of coding, they bought a super-large ring-shaped doughnut.

The perimeter of this doughnut is 1 meter. They like things to be super precise so they divided the doughnut into 10^9 parts, The outermost length of each of them is 1 nanometer. They are numbered from 0 to $10^9 - 1$.

This doughnut was topped with exactly N (N is even) sprinkles. No two sprinkles are in the same part. Khue and Hanh wants to divide the doughnut so that:

- Khue has $5 \cdot 10^8$ consecutive parts. These parts have exactly $N/2$ sprinkles.
- Hanh has $5 \cdot 10^8$ consecutive parts. These parts have exactly $N/2$ sprinkles.

In order to divide, they use an image processing program to count the number of sprinkles in consecutive parts.

You task is to divide the doughnut for Khue and Hanh without using the image processing program too many times. If there are multiple ways to divide the doughnut that satisfy the above conditions, you can answer with any of them.

Interaction

First your program reads the total number of sprinkles N ($1 \le N \le 100\,000$) from the standard input. It is guaranteed that N is even. Then the following process repeats:

- Your program write to the standard output one of the following:
  - `QUERY u v` ($0 \le u, v < 10^9$) — count the number of sprinkles from part u to part v. Please note that: $u > v$ means count the number of sprinkles from part u to $10^9 - 1$ and 0 to v.
  - `YES x` ($0 \le x < 10^9$) — You answer Khue should get parts from x to $(x + 5 \cdot 10^8 - 1) \bmod 10^9$.

- `NO` — You answer there is no division that satisfies Khue and Hanh.

- If your program asks a query, an integer S will be available in the standard input. Your program should then read it.
- Otherwise, your answer will be checked. **Your program should terminate immediately after this.**

You are allowed to interact at most $30 + \left\lfloor \log_2 \sqrt{N} \right\rfloor$ times, including giving an answer.

Communication example

Your output (standard output)	Kattis' input/answer (standard input)	Interpretation
	`4`	There are 4 sprinkles in this doughnut
`QUERY 999999990 10`		You want to count sprinkles in parts from 999 999 990 to 999 999 999 and from 0 to 10
	`4`	There are 4 sprinkles in these parts
`QUERY 0 10`		
	`4`	There are 4 sprinkles in parts 0 to 10
`QUERY 0 5`		You want to count sprinkles in parts 0 to 5
	`3`	There are 3 sprinkles in parts 0 to 5
`QUERY 3 5`		You want to count sprinkles in parts 3 to 5
	`1`	There is 1 sprinkle in parts 3 to 5
`YES 3`		You answer a satisfied division is 3 to 500 000 002

Note

When you write the solution for the interactive problem it is important to keep in mind that if you output some data it is possible that this data is first placed to some internal buffer and may be not directly transferred to the interactor. In order to avoid such situation **you have to use special 'flush' operation each time you output some data**. There are these 'flush' operations in standard libraries of almost all languages. For example, in C++ you may use `fflush(stdout)` or `cout << flush` (it depends on what do you use for output data — `scanf/printf` or `cout`). In Java you can use method 'flush' for output stream, for example, `System.out.flush()`. In Python you can use `stdout.flush()`.

Problem E
Enjoying Elderberries

One day, a flock of hungry birds find a huge tree of elderberries. They want to eat all berries of this tree. These birds quickly alight on some leaves of the tree and eat all berries at where they stand. Now they are thinking of who will eat other berries ...

Since the flock has some *giant* birds and some *tiny* birds, and they observe that the tree has some *big* branches and some *small* branches, fairly distributing these berries among all birds turns out to be a tough problem.

Formally, the tree of elderberries can be presented as a tree with the following properties:

- The tree has n vertices numbered from 1 to n. The root of the tree is vertex 1.
- Each non-leaf vertex (vertex having at least one child) is called a *branch*. A branch is classified as either *big* or *small*. The root is always a **big** branch.
- On each leaf of the tree, there is either one *giant* bird, one *tiny* bird or one elderberry.

The process of determining which bird eats which berry is as below:

- First, every bird and every berry has a *label*. A label is a non-empty string of **at most five** lowercase English characters.
- Second, every bird has a *controlled area*. Consider the bird at vertex v:
  - If it is a **tiny** bird, its *controlled area* is the subtree rooted at vertex p_v, where p_v is the parent of v.
  - If it is a **giant** bird, its *controlled area* is the subtree rooted at vertex b_v, where b_v is the **closest** vertex to v among all **ancestors** of v which are **big** branches.
- Finally, for each berry, the bird who eats it is determined using the following rules:
  - A bird can only eat berries having the same label with it.
  - A bird can only eat berries inside its *controlled area*.
  - If there is more than one bird satisfying the two above conditions, only one bird with **smallest** *controlled area* eats it.

All birds also set some rules for labeling birds and berries:

- A bird or a berry can have same label with other birds and berries. However, no groups of **eight or more** birds have the same label.
- Each berry is inside the *controlled area* of at least one bird with same label.
- No two birds with same label have the same *controlled area*.

- From all of the above, it can be proven that the bird eating every berry can be uniquely determined.

Sadly, *tiny* birds think that these rules give advantages to *giant* birds over them. They claim that, by the rules of setting *controlled areas*, *giant* birds usually control larger areas, which means having more berries. (Although their thought is not always true.) They want their *controlled areas* to be determined using the rules of giant birds. (In other words, they want all tiny birds become giant birds).

Giant birds accept the *tiny* birds' claim. However, they try to change the labels of some birds and berries, so that if we assume **all tiny birds become giant**, the new set of labels still satisfy all the above conditions. Moreover, after changing labels, **no berries have changed owner** (i.e, every berry is still eaten by the same bird).

They want to change as few labels as possible. Please help them!

Input

The first line contains one integer n $(3 \le n \le 150\,000)$ — the number of vertices in the elderberry tree. The i-th of the next n lines describes the i-th vertex in either of the following formats:

- 'p_i t_i', where t_i is either 'B' or 'S': meaning that i is a **branch**. $t_i = B$ and $t_i = S$ mean that i is a *big* or *small* branch, respectively.
- 'p_i T_i L', where T_i is either 'G', 'T' or 'E' and L is a non-empty string of **at most five** English characters: meaning that i is a **leaf**. $t_i = G$, $t_i = T$ and $t_i = E$ mean that in vertex i there is either a *giant* bird, a *tiny* bird or an *elderberry*. L is the label of this bird or berry.

In both formats, p_i satisfies $1 \le p_i < i$ and denotes the parent of vertex i. We assume that $p_1 = 0$.

It is guaranteed that the input describes a valid tree, there is at least one bird and one berry, and all labels satisfy the rules presented above. Remember that **no eight birds** have the same label.

Output

- The first line contains an integer k — the minimum number of labels need changing.
- Each of the next k lines contains an integer x $(1 \le x \le n)$ and a label S — meaning that the label of the bird or berry at vertex x is assigned to S. x should be a leaf of the tree. S should be a non-empty string of at most five lowercase English characters.

If there are multiple optimal solutions, you can output any one of them.

Explanation for examples

Below are figures depicting these three examples. Red vertices are big branches, green vertices are small branches, violet vertices have giant birds, yellow vertices have tiny birds and white vertices have berries.

- In the first example:
  - Initially:
    - There are 3 birds with label 'a' at vertices 6, 7 and 13. Their *controlled areas* are the subtrees rooted at vertices 2, 5 and 1, respectively.
    - There is 1 bird with label 'b' at vertex 12. Its *controlled area* is the subtree rooted at vertex 1.
    - There are 3 elderberries with label 'a' at vertices 3, 8 and 11. They are eaten by the birds at vertices 6, 7 and 13, respectively.
    - There are 2 elderberries with label 'b' at vertices 4 and 9. They are both eaten by the bird at vertex 12.
  - If all tiny birds become giant birds, both the *controlled area* of the birds at vertices 6 and 7 become the subtree rooted at vertex 2, violating the rules (No two birds with same label having same *controlled area*). Hence, at least one bird's label needs changing. We can change the label of the bird at vertex 6 to 'c'. As the bird at vertex 6 eats the berry at vertex 3, we need to change the label of the bird at vertex 3 as well. The solution in which the bird/berry at vertices 7 and 8 get changed is also accepted.
- In the second example:
  - Initially:
    - The *controlled areas* of the birds at vertices 3 and 6 are subtrees rooted at vertices 1 and 5, respectively.
    - The bird at vertex 3 eats the berry at vertex 4.
  - If all tiny birds become giant birds, their *controlled areas* are subtrees rooted at vertices 1 and 2, respectively. As a result, the bird at vertex 6 eats the berry at vertex 4, which is invalid, (The owner of every berry must not change). Hence the label of the bird at vertex 6 must be changed.
- In the third example, no changes are necessary.

Sample Input 1

```
13
0 B
1 B
2 E a
2 E b
2 S
5 G a
5 T a
5 E a
5 E b
1 S
10 E a
10 G b
1 T a
```

Sample Output 1

```
2
3 c
6 c
```

Sample Input 2

```
6
0 B
1 B
1 T a
2 E a
2 S
5 T a
```

Sample Output 2

```
1
6 b
```

Sample Input 3

```
6
0 B
1 G y
1 E y
1 E z
1 T z
1 E z
```

Sample Output 3

```
0
```

Problem F
Fun with Fibonacci

The **Fibonacci sequence** is defined by the following recurrence relation:
$F_n = F_{n-1} + F_{n-2}$,
with seed values $F_0 = 0$ and $F_1 = 1$.
The first few values of the Fibonacci sequence are $0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$
Let:

- $G(1, n)$ denotes the n-th Fibonacci number.
- $G(2, n) = G(1, G(1, n))$,
- $G(i, n) = G(1, G(i - 1, n))$.

Calculate $G(k, n) \bmod p$.

Input

The first line of input contains the only integer T — the number of test cases. $(1 \le T \le 10^5)$.

Each test case consists of one line, containing 3 integers n, k and p, separated by single spaces. $(1 \le k, n \le 10^{18}, 1 \le p \le 10^6)$.

Output

For each test case, print a single line, containing $G(k, n) \bmod p$.

Sample Input 1	Sample Output 1
10 1 1 1000000 1 2 1000000 1 3 1000000 2 1 1000000 2 2 1000000 2 3 1000000 3 1 1000000 3 2 1000000 3 3 1000000 1000000000000000000 1000000000000000000 1000000	1 1 1 1 1 1 2 1 1 890625

Problem G
Grab a Graph

Bash and Chikapu are strongly addicted to... graphs. They enjoy playing with graphs so much that they skip meals quite often. Today, they are going to a supermarket. But instead of grabbing some snack, they grab some graphs.

Bash only likes weighted undirected graph with small numbers of vertices and edges. More precisely, Bash only likes weighted undirected graphs with at most 72 vertices and at most $2\,525$ edges.

Chikapu only likes weighted undirected graph with even fewer edges. More precisely, Chikapu only likes weighted undirected graphs with at most 88 vertices and at most 214 edges.

Bash and Chikapu give Mr. Graph Maker — the owner of the supermarket — an integer A between 0 and 10^{18}, inclusive. They ask him for two graphs where the number of shortest paths from vertex 1 to vertex 2 is exactly A. (Each graph must have at least two vertices, and vertices are numbered starting at 1.) To challenge Mr. Graph Maker a little bit more, both graphs must not contain self-loops or duplicate edges. Of course, the first graph must satisfy Bash's preference and the second graph must satisfy Chikapu's preference (in terms of number of vertices and edges, as above).

Please help Mr. Graph Maker create these graphs.

Notes

Given a graph $G = (V, E)$, a path from vertex u to vertex v is defined as a sequence of edges $(a_0, a_1), (a_1, a_2), \ldots, (a_{k-1}, a_k)$ where:

- $a_0 = u$,
- $a_k = v$,
- $(a_{i-1}, a_i) \in E$.

The weight of a path is the sum of the weights of all edges on that path.

A shortest path from u to v is a path with smallest weight from u to v.

Two paths are considered different iff their sequences of edges are different.

Input

The input contains at most 101 lines. Each line (except the last one) contains an integer A ($0 \le A \le 10^{18}$) representing a test case. The last line contains the number -1, which is not a test case.

Output

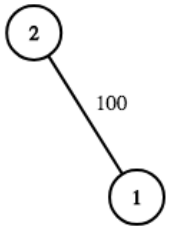
For each test case,

- If it is impossible to create a graph which Bash asks for, print a single line containing the word 'NO'.
- Otherwise, print a line containing the word 'YES'. On the next line, print 2 integers N and M ($2 \le N \le 72, 0 \le M \le 2\,525$) — the number of vertices and edges of Bash's graph. On each of the next M lines, print 3 integers u, v and c, representing an edge connecting vertices u and v with weight c. ($1 \le u, v \le N, 1 \le c \le 10^9$).
- If it is impossible to create a graph which Chikapu asks for, print a single line containing the word 'NO'.
- Otherwise, print a line containing the word 'YES'. On the next line, print 2 integers N and M ($2 \le N \le 88, 0 \le M \le 214$) — the number of vertices and edges of Chikapu's graph. On each of the next M lines, print 3 integers u, v and c, representing an edge connecting vertices u and v with weight c. ($1 \le u, v \le N, 1 \le c \le 10^9$).

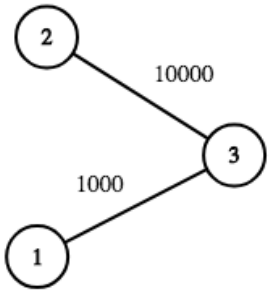
The graphs must not contain any self-loops or duplicate edges, and the number of shortest paths from 1 to 2 must equal A.

Explanation for the first example

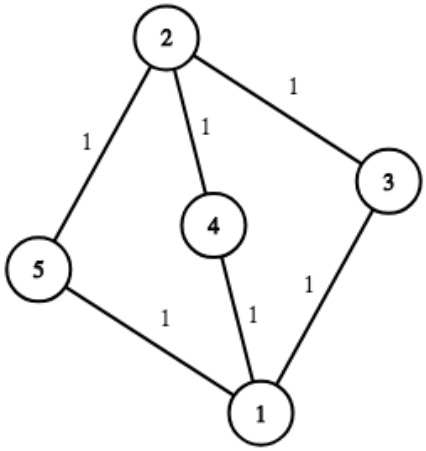
The graph given to Bash in sample 1:



The graph given to Chikapu in sample 1:



The graph given to both Bash and Chikapu in sample 2:



Sample Input 1	Sample Output 1
1 3 -1	YES 2 1 1 2 100 YES 3 2 1 3 1000 3 2 10000 YES 5 6 1 3 1 3 2 1 1 4 1 4 2 1 1 5 1 5 2 1 YES 5 6 1 3 1 3 2 1 1 4 1 4 2 1 1 5 1 5 2 1

Problem H
Hydra's Heads

Princess Perly has been kidnapped by the magical Hydra! The kingdom is in chaos. Now only PyPy — the bravest Knight of the country — can save the day!

Hydra is a powerful magical creature with H heads and T tails. Hydra can breathe fire from each of its heads, and can shoot poison from each of its tails. The only way to kill Hydra is to cut off all H heads and all T tails. Please note that a Hydra with 0 heads and $T > 0$ tails is still alive — in which case it is called a 'Headless Hydra'.

Knight PyPy is indeed brave, but the Knight is still a novice when it comes to fighting magical creatures. Knight PyPy only know 4 moves:

- With the first move, Knight PyPy can cut off exactly one of Hydra's heads.
- With the second move, Knight PyPy can cut off exactly one of Hydra's tails.
- With the third move, Knight PyPy can cut off exactly two of Hydra's heads.
- With the fourth move, Knight PyPy can cut off exactly two of Hydra's tails.

Please remember that even though Hydra is a magical creature, the number of its heads and tails can never be negative. Thus PyPy cannot use a move when there are not enough heads or tails; e.g. PyPy cannot use the first move on a Headless Hydra.

However, Hydra is not so easy to kill. Immediately after PyPy's move,

- If PyPy cuts off exactly one head, a new head grows immediately.
- If PyPy cuts off exactly one tail, two new tails grow immediately.
- If PyPy cuts off exactly two tails, a new head grows immediately.
- If PyPy cuts off exactly two heads, nothing happens.

Note that if after PyPy's move Hydra has 0 heads and 0 tails, new heads and tails can still grow, in which case the Hydra is still alive.

Knight PyPy wants to kill Hydra as soon as possible, so that he can save Princess Perly. What is the minimum number of moves that PyPy needs to use to kill Hydra?

Input

Input contains at most 50 test cases. Each test case contains exactly one line with 2 space-separated integers H and T $(1 \leq H, T \leq 100)$.

The input is terminated by a single line containing two zeros.

Output

For each test case, print exactly one line containing a single integer S:

- If it is impossible to kill Hydra, $S = -1$,
- Otherwise, S is the minimum number of moves to kill Hydra.

Explanation of example

In this test case, Hydra has 3 heads and 3 tails initially. Following is a possible strategy for PyPy, with 9 moves:

- Use the fourth move. Hydra now has 4 heads and 1 tail.
- Use the third move. Hydra now has 2 heads and 1 tail.
- Use the third move. Hydra now has 0 heads and 1 tail.
- Use the second move. Hydra now has 0 heads and 2 tails.
- Use the second move. Hydra now has 0 heads and 3 tails.
- Use the second move. Hydra now has 0 heads and 4 tails.
- Use the fourth move. Hydra now has 1 head and 2 tails.
- Use the fourth move. Hydra now has 2 heads and 0 tails.
- Use the third move. Hydra now has 0 heads and 0 tails. Because PyPy cuts of exactly 2 heads, no new head nor tail grow, and Hydra is dead.

Thus PyPy can kill Hydra with 9 moves. This is also the minimum number of moves for this test case.

Sample Input 1	Sample Output 1
3 3 1 1 0 0	9 3

Problem I
Insider's Identity

In a planet far far away, an intelligence agency plans to send some spies to the Earth to investigate the life there. In order to ensure the secret and secure of this scouting campaign, the agency gives a secret ID to each member.

Each ID is a binary string of length n. To prevent enemies from infiltrating the agency, they choose a pattern P, which is a string of `1` and $\star$, and decides that an ID is valid iff it *satisfies* this pattern P.

A binary string $S = s_1 s_2 \ldots s_n$ *satisfies* a pattern $P = p_1 p_2 \ldots p_m$ iff any of these conditions holds:

- $m = n$, and for each valid index i, either $s_i = 1$ or $p_i = *$.
- $m < n$, and at least one **substring** of S *satisfies* the pattern P.

For example:

- Both strings `101` and `111` satisfy the pattern `1*1`.
- These strings `0101110`, `1110000` and `1010111` satisfy the pattern `1*1`, since `101` and `111` are their substrings.
- The string `0100010` does not satisfy the pattern `1*1`.

The agency wants to know how many spies they can employ, if each member has a unique ID.

Input

- The first line contains one integer n $(1 \le n \le 50)$ — the length of a valid ID.
- The second line contains a string of at most 30 characters `1` and $\star$, where at least half of them are `1`s — the pattern P which all valid IDs must satisfy.

Output

Print a single integer — the maximum number of members in the agency, if each is assigned a unique valid ID.

Sample Input 1	Sample Output 1
10 1	1023

Sample Input 2	Sample Output 2
3 1*1	2

Problem J
Jurassic Jungle

After having read all the books and watched all the movies from the 'Jurassic Park' franchise, Jarvi now knows all about dinosaurs. He has decided to build his own 'Jurassic Park' ... no, parks are too small, Jarvi will build his own 'Jurassic Jungle'.

Jarvi has already started breeding N different dinosaurs. His Jurassic Jungle will have exactly N dinosaur-homes (called dinohomes), numbered from 1 to N, each with exactly one dinosaur.

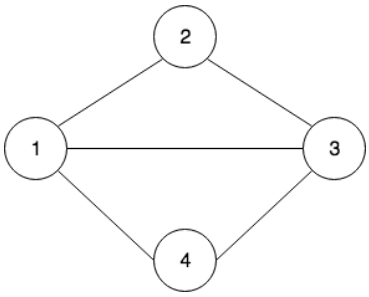
Jarvi knows the secret of keeping dinosaurs happy: every day, each dinosaur must be able to walk around Jurassic Jungle, visiting all other dinohomes, and safely return to his own dinohome before midnight.

To achieve this, Jarvi will build M bidirectional roads, each connecting 2 different dinohomes. Jarvi does not want to build more than one road between any pair of dinohomes.

However, this is not an easy task, as dinosaurs are not good with directions. Starting from his dinohome, a dinosaur will randomly select any dinohome that he has not visited before, and walk there.

The dinosaur will be happy iff no matter how he randomly chooses the path, he must be able to visit all other $N - 1$ other dinohomes, and from the last dinohome he visits he must be able to walk directly to his own dinohome.

Consider the following Jurassic Jungle with $N = 4$ and $M = 5$:



Consider the dinosaur in dinohome numbered 1:

- At the start of his walk, he has only visited dinohome numbered 1 — his own dinohome. Thus he can randomly choose to walk to one of dinohomes 2, 3 or 4.
- If the dinosaur chooses to walk to dinohome 2:
  - He can only continue to walk to dinohome 3, as dinohome 1 was already visited.
  - From dinohome 3, he can only walk to dinohome 4, since dinohomes 1 and 2 were already visited.
  - From dinohome 4, he can directly walk to his own dinohome.
- If the dinosaur chooses to walk to dinohome 3:

– He can randomly choose to walk to either dinohome 2 or 4.
– If he chooses to walk to dinohome 2, he cannot continue his walk, as both dinohomes 1 and 3 were visited.

Thus the dinosaur at dinohome 1 will NOT be happy, since he might not be able to visit all other $N - 1$ dinohomes, depending on how he chooses the path.

Let's also consider the dinosaur in dinohome numbered 2. Starting at his home, he can walk to dinohomes 1, 3 and 4 in this order. At dinohome 4, this dinosaur has visited all other 3 dinohomes, but he cannot directly walk back to his dinohome. Thus the dinosaur at dinohome 2 will NOT be happy.

Please help Jarvi build his Jurassic Jungle so that all his N dinosaurs are happy.

Input

The input contains at most 30 test cases. Each test case contains a single line with exactly 2 space-separated integers N and M. $(3 \leq N \leq 30, 0 \leq M \leq N(N-1)/2)$. The last line of the input contains two -1s and does not represent a test case.

Output

For each test case, if it is impossible to build a Jurassic Jungle satisfying the given conditions, print a single line containing 'NO'.

Otherwise, print a line containing 'YES', followed by M lines. Each line should contain 2 integers u and v, representing a road between the u-th dinohome and the v-th dinohome. Every road should connect two different dinohomes, and every pair of two different dinohomes should be directly connected by at most one road.

Dinohomes are numbered from 1 to N.

Sample Input 1	Sample Output 1
3 3 5 4 -1 -1	YES 1 2 1 3 3 2 NO

Problem K
Kingdom of Kittens

In a country far far away, there are so many cats. The cats quickly dominate the whole area, they can even overwhelm humans! Mature cats plan to establish their own kingdom for their own kittens! The first thing to do is setting the border of their kingdom.

One day, they looked around and found n beautiful bushes. They decided to use these bushes to mark the border of their kingdom.

Unfortunately, cats are not so good at geometry, they do not know any shapes except triangles. Hence, they would like to know if they can make a triangle border containing all of these bushes. The triangle border must have positive area. Please help them!

More formally, given n points on a Cartesian coordinate plane, you should determine if there exists a triangle where all these points are on its sides. Points are allowed to coincide with any of the triangle's vertices. Although all n points have integer coordinates, the triangle's vertices do not need to have integer coordinates.

Input

The input contains multiple test cases. Each test case is described as below:

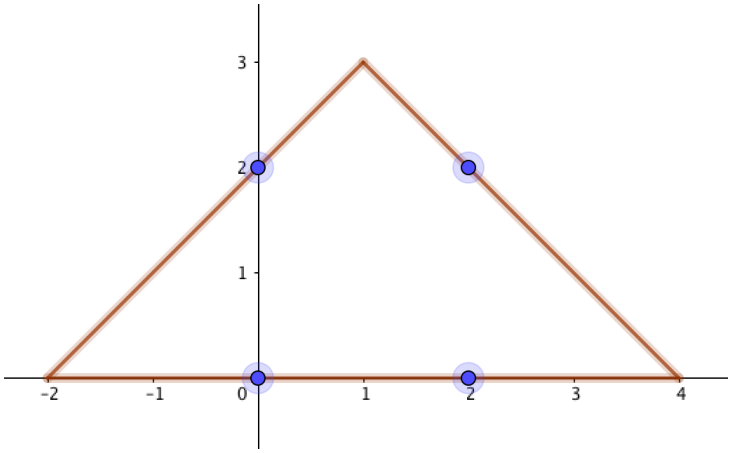
- The first line contains an integer n — the number of points. $(1 \leq n \leq 10^5)$.
- In the next n lines, the i-th line contains two integers x_i and y_i — the coordinates of the i-th point. $(-10^9 \leq x_i, y_i \leq 10^9)$.

The input is terminated with a line containing a single 0.
Sum of N over all test cases in one input file is at most $5 \cdot 10^5$.

Output

For each test case, print 'YES' if there is a triangle satisfying the above constraints. Otherwise, print 'NO'.

Explanation for the first example



Sample Input 1	Sample Output 1
4 0 0 0 2 2 0 2 2 5 0 0 0 2 2 0 2 2 1 1 0	YES NO

Problem L
Lazy Learner

Lang is very lazy at learning English. He is always getting distracted during English classes. Being happy with the sweet message he got from his girlfriend yesterday, Lang invented a game to play during his English class.

Lang has an English dictionary containing n words, denoted as $T_1, T_2, \ldots, T_n$. Denoting as S the message he received, his game is as below:

- First, he chooses two indices ℓ and r $(1 \leq \ell \leq r \leq |S|)$.
- Then, he considers the **substring** of S from ℓ to r, he finds all words in his dictionary which are **subsequences** of this **substring**.
- Next, he sorts all these words in lexicographic order.
- Finally, he chooses a positive integer k and wonders which is the k-th word of this list.

Lang wants to play this game q times. However, he worries that the class would end before he finishes playing, so he asks you to write a program to find the k-th word quickly.

Note

- Given a string $S = s_1 s_2 \ldots s_n$:
  - The **substring** of S from ℓ to r $(1 \leq \ell \leq r \leq n)$ is the string $s_\ell s_{\ell+1} \ldots s_r$.
  - A string $T = t_1 t_2 \ldots t_m$ is a **subsequence** of S iff there exists a sequence of indices $1 \leq i_1 < i_2 < \cdots < i_m \leq n$ such that $S_{i_j} = T_j \; \forall 1 \leq j \leq m$.
- A string $S = s_1 s_2 \ldots s_n$ is lexicographically smaller than a string $T = t_1 t_2 \ldots t_m$ iff any of the following is satisfied:
  - $n < m$ and $S_i = T_i \; \forall 1 \leq i \leq n$.
  - There exists an index j such that $1 \leq j \leq \min(m, n), S_i = T_i \; \forall 1 \leq i < j$ and $S_j < T_j$.

Input

- The first line contains a string S of at most 500 lowercase English characters — the message Lang received yesterday.
- The second line contains two integers n $(1 \leq n \leq 2 \cdot 10^4)$ — the number of words in Lang's dictionary, and q $(1 \leq q \leq 3 \cdot 10^5)$ — the number of times Lang plays his game.
- Each of the next n lines contain a non-empty string of lowercase English characters — a word in Lang's dictionary. The total length of these n words does not exceed $8 \cdot 10^4$.
- Each of the last q lines contain three positive integers ℓ, r, and k $(1 \leq \ell \leq r \leq |S|, 1 \leq k \leq n)$ presenting a game as described above.

Output

Print q lines describing the words Lang is looking for in these q games. **If the length of the word is longer than 10 characters, print only the first 10 characters.** If in some game, that word does not exist (i.e, k is greater than the number of words in the list), print 'NO SUCH WORD' instead.

Please note that the checker of this problem is case-sensitive.

Sample Input 1	Sample Output 1
abcd 3 5 a ac bd 1 3 1 1 3 2 1 3 3 2 4 1 2 4 2	a ac NO SUCH WORD bd NO SUCH WORD