A. Cutting Banner

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

A large banner with word CODEFORCES was ordered for the 1000-th onsite round of Codeforces$^\omega$ that takes place on the Miami beach. Unfortunately, the company that made the banner mixed up two orders and delivered somebody else's banner that contains someone else's word. The word on the banner consists only of upper-case English letters.

There is very little time to correct the mistake. All that we can manage to do is to cut out some *substring* from the banner, i.e. several consecutive letters. After that all the resulting parts of the banner will be glued into a single piece (if the beginning or the end of the original banner was cut out, only one part remains); it is not allowed change the relative order of parts of the banner (i.e. after a substring is cut, several first and last letters are left, it is allowed only to glue the last letters to the right of the first letters). Thus, for example, for example, you can cut a substring out from string 'TEMPLATE' and get string 'TEMPLE' (if you cut out string AT), 'PLATE' (if you cut out TEM), 'T' (if you cut out EMPLATE), etc.

Help the organizers of the round determine whether it is possible to cut out of the banner some substring in such a way that the remaining parts formed word CODEFORCES.

Input

The single line of the input contains the word written on the banner. The word only consists of upper-case English letters. The word is non-empty and its length doesn't exceed 100 characters. It is guaranteed that the word isn't word CODEFORCES.

Output

Print 'YES', if there exists a way to cut out the substring, and 'NO' otherwise (without the quotes).

Sample test(s)

input
```
CODEWAITFORITFORCES
```
output
```
YES
```

input
```
BOTTOMCODER
```
output
```
NO
```

input
```
DECODEFORCES
```
output
```
YES
```

input
```
DOGEFORCES
```
output
```
NO
```

B. Quasi Binary

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

A number is called *quasibinary* if its decimal representation contains only digits 0 or 1. For example, numbers 0, 1, 101, 110011 — are quasibinary and numbers 2, 12, 900 are not.

You are given a positive integer n. Represent it as a sum of minimum number of quasibinary numbers.

Input

The first line contains a single integer n $(1 \le n \le 10^6)$.

Output

In the first line print a single integer k — the minimum number of numbers in the representation of number n as a sum of quasibinary numbers.

In the second line print k numbers — the elements of the sum. All these numbers should be quasibinary according to the definition above, their sum should equal n. Do not have to print the leading zeroes in the numbers. The order of numbers doesn't matter. If there are multiple possible representations, you are allowed to print any of them.

Sample test(s)

input

```
9
```

output

```
9
1 1 1 1 1 1 1 1 1
```

input

```
32
```

output

```
3
10 11 11
```

C. Tourist's Notes

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

A tourist hiked along the mountain range. The hike lasted for n days, during each day the tourist noted height above the sea level. On the i-th day height was equal to some integer h_i. The tourist pick smooth enough route for his hike, meaning that the between any two consecutive days height changes by at most 1, i.e. for all i's from 1 to $n - 1$ the inequality $|h_i - h_{i+1}| \le 1$ holds.

At the end of the route the tourist rafted down a mountain river and some notes in the journal were washed away. Moreover, the numbers in the notes could have been distorted. Now the tourist wonders what could be the maximum height during his hike. Help him restore the maximum possible value of the maximum height throughout the hike or determine that the notes were so much distorted that they do not represent any possible height values that meet limits $|h_i - h_{i+1}| \le 1$.

Input

The first line contains two space-separated numbers, n and m $(1 \le n \le 10^8, 1 \le m \le 10^5)$ — the number of days of the hike and the number of notes left in the journal.

Next m lines contain two space-separated integers d_i and h_{d_i} $(1 \le d_i \le n, 0 \le h_{d_i} \le 10^8)$ — the number of the day when the i-th note was made and height on the d_i-th day. It is guaranteed that the notes are given in the chronological order, i.e. for all i from 1 to $m - 1$ the following condition holds: $d_i < d_{i+1}$.

Output

If the notes aren't contradictory, print a single integer — the maximum possible height value throughout the whole route.

If the notes do not correspond to any set of heights, print a single word '`IMPOSSIBLE`' (without the quotes).

Sample test(s)

input

```
8 2
2 0
7 0
```

output

```
2
```

input

```
8 3
2 0
7 0
8 3
```

output

```
IMPOSSIBLE
```

Note

For the first sample, an example of a correct height sequence with a maximum of 2: $(0, 0, 1, 2, 1, 1, 0, 1)$.

In the second sample the inequality between h_7 and h_8 does not hold, thus the information is inconsistent.

D. Weird Chess

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

Igor has been into chess for a long time and now he is sick of the game by the ordinary rules. He is going to think of new rules of the game and become world famous.

Igor's chessboard is a square of size $n \times n$ cells. Igor decided that simple rules guarantee success, that's why his game will have only one type of pieces. Besides, all pieces in his game are of the same color. The possible moves of a piece are described by a set of *shift vectors*. The next passage contains a formal description of available moves.

Let the rows of the board be numbered from top to bottom and the columns be numbered from left to right from 1 to n. Let's assign to each square a pair of integers (x, y) — the number of the corresponding column and row. Each of the possible moves of the piece is defined by a pair of integers (dx, dy); using this move, the piece moves from the field (x, y) to the field $(x + dx, y + dy)$. You can perform the move if the cell $(x + dx, y + dy)$ is within the boundaries of the board and doesn't contain another piece. Pieces that stand on the cells other than (x, y) and $(x + dx, y + dy)$ are not important when considering the possibility of making the given move (for example, like when a knight moves in usual chess).

Igor offers you to find out what moves his chess piece can make. He placed several pieces on the board and for each unoccupied square he told you whether it is attacked by any present piece (i.e. whether some of the pieces on the field can move to that cell). Restore a possible set of shift vectors of the piece, or else determine that Igor has made a mistake and such situation is impossible for any set of shift vectors.

Input

The first line contains a single integer n ($1 \le n \le 50$).

The next n lines contain n characters each describing the position offered by Igor. The j-th character of the i-th string can have the following values:

- `o` — in this case the field (i, j) is occupied by a piece and the field may or may not be attacked by some other piece;
- `x` — in this case field (i, j) is attacked by some piece;
- `.` — in this case field (i, j) isn't attacked by any piece.

It is guaranteed that there is at least one piece on the board.

Output

If there is a valid set of moves, in the first line print a single word '`YES`' (without the quotes). Next, print the description of the set of moves of a piece in the form of a $(2n - 1) \times (2n - 1)$ board, the center of the board has a piece and symbols '`x`' mark cells that are attacked by it, in a format similar to the input. See examples of the output for a full understanding of the format. If there are several possible answers, print any of them.

If a valid set of moves does not exist, print a single word '`NO`'.

Sample test(s)

input

```
5
oxxxx
x...x
x...x
x...x
xxxxo
```

output

```
YES
....x....
....x....
....x....
....x....
xxxxoxxxx
....x....
....x....
....x....
....x....
```

input

```
6
.x.x..
x.x.x.
.xo..x
x..ox.
.x.x.x
..x.x.
```

output

```
YES
...........
...........
...........
....x.x....
...x...x...
```

```
.....o.....
...x...x...
....x.x....
...........
...........
...........
```

input

```
3
o.x
oxx
o.x
```

output

```
NO
```

Note

In the first sample test the piece is a usual chess rook, and in the second sample test the piece is a usual chess knight.

E. Demiurges Play Again

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

Demiurges Shambambukli and Mazukta love to watch the games of ordinary people. Today, they noticed two men who play the following game.

There is a rooted tree on n nodes, m of which are *leaves* (a leaf is a nodes that does not have any children), edges of the tree are directed from parent to children. In the leaves of the tree integers from 1 to m are placed in such a way that each number appears exactly in one leaf.

Initially, the root of the tree contains a piece. Two players move this piece in turns, during a move a player moves the piece from its current nodes to one of its children; if the player can not make a move, the game ends immediately. The *result* of the game is the number placed in the leaf where a piece has completed its movement. The player who makes the first move tries to maximize the result of the game and the second player, on the contrary, tries to minimize the result. We can assume that both players move optimally well.

Demiurges are omnipotent, so before the game they can arbitrarily rearrange the numbers placed in the leaves. Shambambukli wants to rearrange numbers so that the result of the game when both players play optimally well is as large as possible, and Mazukta wants the result to be as small as possible. What will be the outcome of the game, if the numbers are rearranged by Shambambukli, and what will it be if the numbers are rearranged by Mazukta? Of course, the Demiurges choose the best possible option of arranging numbers.

Input

The first line contains a single integer n — the number of nodes in the tree ($1 \le n \le 2 \cdot 10^5$).

Each of the next $n - 1$ lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$) — the ends of the edge of the tree; the edge leads from node u_i to node v_i. It is guaranteed that the described graph is a rooted tree, and the root is the node 1.

Output

Print two space-separated integers — the maximum possible and the minimum possible result of the game.

Sample test(s)

input

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

output

```
3 2
```

input

```
6
1 2
1 3
3 4
1 5
5 6
```

output

```
3 3
```

Note

Consider the first sample. The tree contains three leaves: 3, 4 and 5. If we put the maximum number 3 at node 3, then the first player moves there and the result will be 3. On the other hand, it is easy to see that for any rearrangement the first player can guarantee the result of at least 2.

In the second sample no matter what the arragment is the first player can go along the path that ends with a leaf with number 3.

F. A Heap of Heaps

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Andrew skipped lessons on the subject 'Algorithms and Data Structures' for the entire term. When he came to the final test, the teacher decided to give him a difficult task as a punishment.

The teacher gave Andrew an array of n numbers $a_1, \ldots, a_n$. After that he asked Andrew for each k from 1 to $n - 1$ to build a k-ary heap on the array and count the number of elements for which the property of the minimum-rooted heap is violated, i.e. the value of an element is less than the value of its parent.

Andrew looked up on the Wikipedia that a k-ary heap is a rooted tree with vertices in elements of the array. If the elements of the array are indexed from 1 to n, then the children of element v are elements with indices $k(v - 1) + 2, \ldots, kv + 1$ (if some of these elements lie outside the borders of the array, the corresponding children are absent). In any k-ary heap every element except for the first one has exactly one parent; for the element 1 the parent is absent (this element is the *root* of the heap). Denote $p(v)$ as the number of the parent of the element with the number v. Let's say that for a non-root element v the *property of the heap is violated* if $a_v < a_{p(v)}$.

Help Andrew cope with the task!

Input

The first line contains a single integer n $(2 \le n \le 2 \cdot 10^5)$.

The second line contains n space-separated integers $a_1, \ldots, a_n$ $(-10^9 \le a_i \le 10^9)$.

Output

in a single line print $n - 1$ integers, separate the consecutive numbers with a single space — the number of elements for which the property of the k-ary heap is violated, for $k = 1, 2, \ldots, n - 1$.

Sample test(s)

input

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

output

```
3 2 1 0
```

input

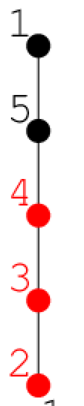
```
6
2 2 2 2 2 2
```

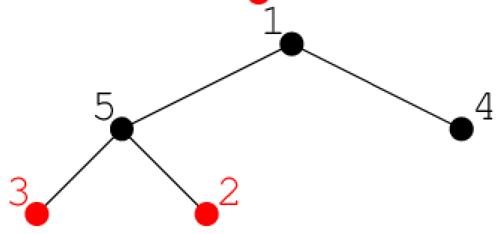
output

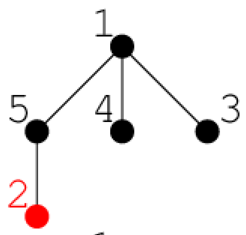
```
0 0 0 0 0
```

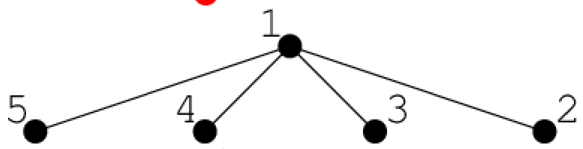
Note

Pictures with the heaps for the first sample are given below; elements for which the property of the heap is violated are marked with red.









In the second sample all elements are equal, so the property holds for all pairs.

G. Berserk Robot

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

Help! A robot escaped our lab and we need help finding it.

The lab is at the point $(0, 0)$ of the coordinate plane, at time 0 the robot was there. The robot's movements are defined by a program — a string of length l, consisting of characters `U`, `L`, `D`, `R`. Each second the robot executes the next command in his program: if the current coordinates of the robot are (x, y), then commands `U`, `L`, `D`, `R` move it to cells $(x, y + 1)$, $(x - 1, y)$, $(x, y - 1)$, $(x + 1, y)$ respectively. The execution of the program started at time 0. The program is looped, i.e. each l seconds of executing the program start again from the first character. Unfortunately, we don't know what program was loaded into the robot when he left the lab.

Our radars managed to find out the position of the robot at n moments of time: we know that at the moment of time t_i the robot is at the point (x_i, y_i). Given this data, either help to determine what program could be loaded into the robot, or determine that no possible program meets the data and the robot must have broken down.

Input

The first line of the input contains two space-separated integers n and l ($1 \le n \le 2 \cdot 10^5$, $1 \le l \le 2 \cdot 10^6$).

Next n lines contain three space-separated integers — t_i, x_i, y_i ($1 \le t_i \le 10^{18}$, $-10^{18} \le x_i, y_i \le 10^{18}$). The radar data is given chronologically, i.e. $t_i < t_{i+1}$ for all i from 1 to $n - 1$.

Output

Print any of the possible programs that meet the data. If no program meets the data, print a single word '`NO`' (without the quotes).

Sample test(s)

input

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

output

```
RDL
```

input

```
2 2
1 1 0
999 1 0
```

output

```
RL
```

input

```
2 5
10 10 0
20 0 0
```

output

```
NO
```

H. Summer Dichotomy

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

T students applied into the ZPP class of Summer Irrelevant School. The organizing committee of the school may enroll any number of them, but at least t students must be enrolled. The enrolled students should be divided into two groups in any manner (it is possible that one of the groups will be empty!)

During a shift the students from the ZPP grade are tutored by n teachers. Due to the nature of the educational process, each of the teachers should be assigned to exactly one of two groups (it is possible that no teacher will be assigned to some of the groups!). The i-th teacher is willing to work in a group as long as the group will have at least l_i and at most r_i students (otherwise it would be either too boring or too hard). Besides, some pairs of the teachers don't like each other other and therefore can not work in the same group; in total there are m pairs of conflicting teachers.

You, as the head teacher of Summer Irrelevant School, have got a difficult task: to determine how many students to enroll in each of the groups and in which group each teacher will teach.

Input

The first line contains two space-separated integers, t and T ($1 \le t \le T \le 10^9$).

The second line contains two space-separated integers n and m ($1 \le n \le 10^5$, $0 \le m \le 10^5$).

The i-th of the next n lines contain integers l_i and r_i ($0 \le l_i \le r_i \le 10^9$).

The next m lines describe the pairs of conflicting teachers. Each of these lines contain two space-separated integers — the indices of teachers in the pair. The teachers are indexed starting from one. It is guaranteed that no teacher has a conflict with himself and no pair of conflicting teachers occurs in the list more than once.

Output

If the distribution is possible, print in the first line a single word '`POSSIBLE`' (without the quotes). In the second line print two space-separated integers n_1 and n_2 — the number of students in the first and second group, correspondingly, the contstraint $t \le n_1 + n_2 \le T$ should be met. In the third line print n characters, the i-th of which should be 1 or 2, if the i-th teacher should be assigned to the first or second group, correspondingly. If there are multiple possible distributions of students and teachers in groups, you can print any of them.

If the sought distribution doesn't exist, print a single word '`IMPOSSIBLE`' (without the quotes).

Sample test(s)

input

```
10 20
3 0
3 6
4 9
16 25
```

output

```
POSSIBLE
4 16
112
```

input

```
1 10
3 3
0 10
0 10
0 10
1 2
1 3
2 3
```

output

```
IMPOSSIBLE
```