Sponsored by Telegram

Graphs, DFS and Applications

A. New Year Transportation

time limit per test: 2 seconds
memory limit per test: 1024 MB
input: standard input
output: standard output

New Year is coming in Line World! In this world, there are n cells numbered by integers from 1 to n, as a $1 \times n$ board. People live in cells. However, it was hard to move between distinct cells, because of the difficulty of escaping the cell. People wanted to meet people who live in other cells.

So, user tncks0121 has made a transportation system to move between these cells, to celebrate the New Year. First, he thought of $n - 1$ positive integers $a_1, a_2, ..., a_{n-1}$. For every integer i where $1 \le i \le n - 1$ the condition $1 \le a_i \le n - i$ holds. Next, he made $n - 1$ portals, numbered by integers from 1 to $n - 1$. The i-th $(1 \le i \le n - 1)$ portal connects cell i and cell $(i + a_i)$, and one can travel from cell i to cell $(i + a_i)$ using the i-th portal. Unfortunately, one cannot use the portal backwards, which means one cannot move from cell $(i + a_i)$ to cell i using the i-th portal. It is easy to see that because of condition $1 \le a_i \le n - i$ one can't leave the Line World using portals.

Currently, I am standing at cell 1, and I want to go to cell t. However, I don't know whether it is possible to go there. Please determine whether I can go to cell t by only using the construted transportation system.

Input

The first line contains two space-separated integers n $(3 \le n \le 3 \times 10^4)$ and t $(2 \le t \le n)$ — the number of cells, and the index of the cell which I want to go to.

The second line contains $n - 1$ space-separated integers $a_1, a_2, ..., a_{n-1}$ $(1 \le a_i \le n - i)$. It is guaranteed, that using the given transportation system, one cannot leave the Line World.

Output

If I can go to cell t using the transportation system, print "YES". Otherwise, print "NO".

Examples

input

```
8 4
1 2 1 2 1 2 1
```

output

```
YES
```

input

```
8 5
1 2 1 2 1 1 1
```

output

```
NO
```

Note

In the first sample, the visited cells are: $1, 2, 4$; so we can successfully visit the cell 4.

In the second sample, the possible cells to visit are: $1, 2, 4, 6, 7, 8$; so we can't visit the cell 5, which we want to visit.

B. DZY Loves Chessboard

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

DZY loves chessboard, and he enjoys playing with it.

He has a chessboard of n rows and m columns. Some cells of the chessboard are bad, others are good. For every good cell, DZY wants to put a chessman on it. Each chessman is either white or black. After putting all chessmen, DZY wants that no two chessmen with the same color are on two adjacent cells. Two cells are adjacent if and only if they share a common edge.

You task is to find any suitable placement of chessmen on the given chessboard.

Input

The first line contains two space-separated integers n and m $(1 \le n, m \le 100)$.

Each of the next n lines contains a string of m characters: the j-th character of the i-th string is either "." or "-". A "." means that the corresponding cell (in the i-th row and the j-th column) is good, while a "-" means it is bad.

Output

Output must contain n lines, each line must contain a string of m characters. The j-th character of the i-th string should be either "W", "B" or "-". Character "W" means the chessman on the cell is white, "B" means it is black, "-" means the cell is a bad cell.

If multiple answers exist, print any of them. It is guaranteed that at least one answer exists.

Examples

input

```
1 1
.
```

output

```
B
```

input

```
2 2
..
..
```

output

```
BW
WB
```

input

```
3 3
.-.
---
--.
```

output

```
B-B
---
--B
```

Note

In the first sample, DZY puts a single black chessman. Of course putting a white one is also OK.

In the second sample, all 4 cells are good. No two same chessmen share an edge in the sample output.

In the third sample, no good cells are adjacent. So you can just put 3 chessmen, no matter what their colors are.

C. Party

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A company has n employees numbered from 1 to n. Each employee either has no immediate manager or exactly one immediate manager, who is another employee with a different number. An employee A is said to be the *superior* of another employee B if at least one of the following is true:

- Employee A is the immediate manager of employee B
- Employee B has an immediate manager employee C such that employee A is the superior of employee C.

The company will not have a managerial cycle. That is, there will not exist an employee who is the superior of his/her own immediate manager.

Today the company is going to arrange a party. This involves dividing all n employees into several groups: every employee must belong to exactly one group. Furthermore, within any single group, there must not be two employees A and B such that A is the superior of B.

What is the minimum number of groups that must be formed?

Input

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

The next n lines contain the integers p_i $(1 \le p_i \le n$ or $p_i =$ -1). Every p_i denotes the immediate manager for the i-th employee. If p_i is -1, that means that the i-th employee does not have an immediate manager.

It is guaranteed, that no employee will be the immediate manager of him/herself ($p_i \ne i$). Also, there will be no managerial cycles.

Output

Print a single integer denoting the minimum number of groups that will be formed in the party.

Examples

input

```
5
-1
1
2
1
-1
```

output

```
3
```

Note

For the first example, three groups are sufficient, for example:

- Employee 1
- Employees 2 and 4
- Employees 3 and 5

D. NP-Hard Problem

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Recently, Pari and Arya did some research about NP-Hard problems and they found the *minimum vertex cover* problem very interesting.

Suppose the graph G is given. Subset A of its vertices is called a *vertex cover* of this graph, if for each edge uv there is at least one endpoint of it in this set, i.e. $u \in A$ or $v \in A$ (or both).

Pari and Arya have won a great undirected graph as an award in a team contest. Now they have to split it in two parts, but both of them want their parts of the graph to be a vertex cover.

They have agreed to give you their graph and you need to find two **disjoint** subsets of its vertices A and B, such that both A and B are vertex cover or claim it's impossible. Each vertex should be given to no more than one of the friends (or you can even keep it for yourself).

Input

The first line of the input contains two integers n and m $(2 \le n \le 100\,000,\ 1 \le m \le 100\,000)$ — the number of vertices and the number of edges in the prize graph, respectively.

Each of the next m lines contains a pair of integers u_i and v_i $(1 \le u_i,\ v_i \le n)$, denoting an undirected edge between u_i and v_i. It's guaranteed the graph won't contain any self-loops or multiple edges.

Output

If it's impossible to split the graph between Pari and Arya as they expect, print "`-1`" (without quotes).

If there are two disjoint sets of vertices, such that both sets are vertex cover, print their descriptions. Each description must contain two lines. The first line contains a single integer k denoting the number of vertices in that vertex cover, and the second line contains k integers — the indices of vertices. Note that because of $m \ge 1$, vertex cover cannot be empty.

Examples

input

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

output

```
1
2
2
1 3
```

input

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

output

```
-1
```

Note

In the first sample, you can give the vertex number 2 to Arya and vertices numbered 1 and 3 to Pari and keep vertex number 4 for yourself (or give it someone, if you wish).

In the second sample, there is no way to satisfy both Pari and Arya.

E. Reposts

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

One day Polycarp published a funny picture in a social network making a poll about the color of his handle. Many of his friends started reposting Polycarp's joke to their news feed. Some of them reposted the reposts and so on.

These events are given as a sequence of strings "*name1* `reposted` *name2*", where *name1* is the name of the person who reposted the joke, and *name2* is the name of the person from whose news feed the joke was reposted. It is guaranteed that for each string "*name1* `reposted` *name2*" user "*name1*" didn't have the joke in his feed yet, and "*name2*" already had it in his feed by the moment of repost. Polycarp was registered as "`Polycarp`" and initially the joke was only in his feed.

Polycarp measures the popularity of the joke as the length of the largest repost chain. Print the popularity of Polycarp's joke.

Input

The first line of the input contains integer n $(1 \le n \le 200)$ — the number of reposts. Next follow the reposts in the order they were made. Each of them is written on a single line and looks as "*name1* `reposted` *name2*". All the names in the input consist of lowercase or uppercase English letters and/or digits and have lengths from 2 to 24 characters, inclusive.

We know that the user names are case-insensitive, that is, two names that only differ in the letter case correspond to the same social network user.

Output

Print a single integer — the maximum length of a repost chain.

Examples

input

```
5
tourist reposted Polycarp
Petr reposted Tourist
WJMZBMR reposted Petr
sdya reposted wjmzbmr
vepifanov reposted sdya
```

output

```
6
```

input

```
6
Mike reposted Polycarp
Max reposted Polycarp
EveryOne reposted Polycarp
111 reposted Polycarp
VkCup reposted Polycarp
Codeforces reposted Polycarp
```

output

```
2
```

input

```
1
SoMeStRaNgEgUe reposted PoLyCaRp
```

output

```
2
```

F. Journey

time limit per test: 3 seconds

memory limit per test: 1024 MB

input: standard input

output: standard output

Recently Irina arrived to one of the most famous cities of Berland — the Berlatov city. There are n showplaces in the city, numbered from 1 to n, and some of them are connected by one-directional roads. The roads in Berlatov are designed in a way such that there **are no** cyclic routes between showplaces.

Initially Irina stands at the showplace 1, and the endpoint of her journey is the showplace n. Naturally, Irina wants to visit as much showplaces as she can during her journey. However, Irina's stay in Berlatov is limited and she can't be there for more than T time units.

Help Irina determine how many showplaces she may visit during her journey from showplace 1 to showplace n within a time not exceeding T. It is guaranteed that there is at least one route from showplace 1 to showplace n such that Irina will spend no more than T time units passing it.

Input

The first line of the input contains three integers n, m and T ($2 \le n \le 5000,\ 1 \le m \le 5000,\ 1 \le T \le 10^9$) — the number of showplaces, the number of roads between them and the time of Irina's stay in Berlatov respectively.

The next m lines describes roads in Berlatov. i-th of them contains 3 integers u_i, v_i, t_i ($1 \le u_i, v_i \le n, u_i \ne v_i, 1 \le t_i \le 10^9$), meaning that there is a road starting from showplace u_i and leading to showplace v_i, and Irina spends t_i time units to pass it. It is guaranteed that the roads do not form cyclic routes.

It is guaranteed, that there is at most one road between each pair of showplaces.

Output

Print the single integer k ($2 \le k \le n$) — the maximum number of showplaces that Irina can visit during her journey from showplace 1 to showplace n within time not exceeding T, in the first line.

Print k distinct integers in the second line — indices of showplaces that Irina will visit on her route, in the order of encountering them.

If there are multiple answers, print any of them.

Examples

input

```
4 3 13
1 2 5
2 3 7
2 4 8
```

output

```
3
1 2 4
```

input

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

output

```
4
1 2 4 6
```

input

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

output

```
3
1 3 5
```

Codeforces (c) Copyright 2010-2017 Mike Mirzayanov
The only programming contests Web 2.0 platform