A. Dice Tower

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

A dice is a cube, its faces contain distinct integers from 1 to 6 as black points. The sum of numbers at the opposite dice faces always equals 7. Please note that there are only two dice (these dices are mirror of each other) that satisfy the given constraints (both of them are shown on the picture on the left).

Alice and Bob play dice. Alice has built a tower from n dice. We know that in this tower the adjacent dice contact with faces with distinct numbers. Bob wants to uniquely identify the numbers written on the faces of all dice, from which the tower is built. Unfortunately, Bob is looking at the tower from the face, and so he does not see all the numbers on the faces. Bob sees the number on the top of the tower and the numbers on the two adjacent sides (on the right side of the picture shown what Bob sees).

Help Bob, tell whether it is possible to uniquely identify the numbers on the faces of all the dice in the tower, or not.

Input

The first line contains a single integer n ($1 \le n \le 100$) — the number of dice in the tower.

The second line contains an integer x ($1 \le x \le 6$) — the number Bob sees at the top of the tower. Next n lines contain two space-separated integers each: the i-th line contains numbers a_i, b_i ($1 \le a_i, b_i \le 6$; $a_i \ne b_i$) — the numbers Bob sees on the two sidelong faces of the i-th dice in the tower.

Consider the dice in the tower indexed from top to bottom from 1 to n. That is, the topmost dice has index 1 (the dice whose top face Bob can see). It is guaranteed that it is possible to make a dice tower that will look as described in the input.

Output

Print "YES" (without the quotes), if it is possible to to uniquely identify the numbers on the faces of all the dice in the tower. If it is impossible, print "NO" (without the quotes).

Examples

input

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

output

```
YES
```

input

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

output

```
NO
```

B. Well-known Numbers

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

Numbers k-bonacci (k is integer, $k > 1$) are a generalization of Fibonacci numbers and are determined as follows:

- $F(k, n) = 0$, for integer n, $1 \le n < k$;
- $F(k, k) = 1$;
- $F(k, n) = F(k, n - 1) + F(k, n - 2) + \ldots + F(k, n - k)$, for integer n, $n > k$.

Note that we determine the k-bonacci numbers, $F(k, n)$, only for integer values of n and k.

You've got a number s, represent it as a sum of several (at least two) **distinct** k-bonacci numbers.

Input

The first line contains two integers s and k ($1 \le s, k \le 10^9$; $k > 1$).

Output

In the first line print an integer m ($m \ge 2$) that shows how many numbers are in the found representation. In the second line print m distinct integers $a_1, a_2, \ldots, a_m$. Each printed integer should be a k-bonacci number. The sum of printed integers must equal s.

It is guaranteed that the answer exists. If there are several possible answers, print any of them.

Examples

input

```
5 2
```

output

```
3
0 2 3
```

input

```
21 5
```

output

```
3
4 1 16
```

C. Barcode

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

You've got an $n \times m$ pixel picture. Each pixel can be white or black. Your task is to change the colors of as few pixels as possible to obtain a barcode picture.

A picture is a barcode if the following conditions are fulfilled:

- All pixels in each column are of the same color.
- The width of each monochrome vertical line is at least x and at most y pixels. In other words, if we group all neighbouring columns of the pixels with equal color, the size of each group can not be less than x or greater than y.

Input

The first line contains four space-separated integers n, m, x and y ($1 \le n, m, x, y \le 1000$; $x \le y$).

Then follow n lines, describing the original image. Each of these lines contains exactly m characters. Character "`.`" represents a white pixel and "#" represents a black pixel. The picture description doesn't have any other characters besides "`.`" and "#".

Output

In the first line print the minimum number of pixels to repaint. It is guaranteed that the answer exists.

Examples

input

```
6 5 1 2
##.#.
.###.
###..
#...#
.##.#
###..
```

output

```
11
```

input

```
2 5 1 1
#####
.....
```

output

```
5
```

Note

In the first test sample the picture after changing some colors can looks as follows:

```
.##..
.##..
.##..
.##..
.##..
.##..
```

In the second test sample the picture after changing some colors can looks as follows:

```
.#.#.
.#.#.
```

D. Snake

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let us remind you the rules of a very popular game called "Snake" (or sometimes "Boa", "Python" or "Worm").

The game field is represented by an $n \times m$ rectangular table. Some squares of the field are considered impassable (walls), all other squares of the fields are passable.

You control a snake, the snake consists of segments. Each segment takes up exactly one passable square of the field, but any passable square contains at most one segment. All segments are indexed by integers from 1 to k, where k is the snake's length. The 1-th segment is the head and the k-th segment is the tail. For any i ($1 \le i < k$), segments with indexes i and $i + 1$ are located in the adjacent squares of the field, that is, these squares share a common side.

One of the passable field squares contains an apple. The snake's aim is to reach the apple and eat it (that is, to position its head in the square with the apple).

The snake moves throughout the game. During one move the snake can move its head to an adjacent field square. All other segments follow the head. That is, each segment number i $(1 < i \le k)$ moves to the square that has just had segment number $i - 1$. Consider that all segments including the head move simultaneously (see the second test sample). If the snake's head moves to an unpassable square or to the square, occupied by its other segment, the snake dies. That's why we will consider such moves unvalid.

Your task is to determine the minimum number of valid moves that the snake needs to reach the apple.

Input

The first line contains two space-separated integers n and m ($1 \le n, m \le 15$) — the number of rows and columns of the game field.

Next n lines describe the game field. Each of these lines contains m characters. Character "#" represents a wall, "." is a passable square, "@" is an apple. The snake's first segment is represented by character "1", the second one segment — by character "2" and so on.

The game field description doesn't contain any characters besides "#', ".", "@" and digits (except 0). It is guaranteed that the described field is correct. It is guaranteed that the described field contains exactly one apple and exactly one snake, the snake's length is at least 3 and at most 9.

Output

Print a single integer to the output — the minimum number of moves needed to reach the apple. If the snake can't reach the apple, print -1.

Examples

input

```
4 5
##...
..1#@
432#.
...#.
```

output

```
4
```

input

```
4 4
#78#
.612
.543
..@.
```

output

```
6
```

input

```
3 2
3@
2#
1#
```

output

```
-1
```

E. Unsolvable

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

Consider the following equation:

$$z = \left[\frac{x}{2}\right] + y + xy,$$

where sign $[a]$ represents the integer part of number a.
Let's find all integer z $(z > 0)$, for which this equation is *unsolvable* in positive integers. The phrase "unsolvable in positive integers" means that there are no such positive integers x and y $(x, y > 0)$, for which the given above equation holds.

Let's write out all such z in the increasing order: z_1, z_2, z_3, and so on $(z_i < z_{i+1})$. Your task is: given the number n, find the number z_n.

Input

The first line contains a single integer n ($1 \le n \le 40$).

Output

Print a single integer — the number z_n modulo 1000000007 $(10^9 + 7)$. It is guaranteed that the answer exists.

Examples

input

```
1
```

output

```
1
```

input

```
2
```

output

```
3
```

input

```
3
```

output

```
15
```