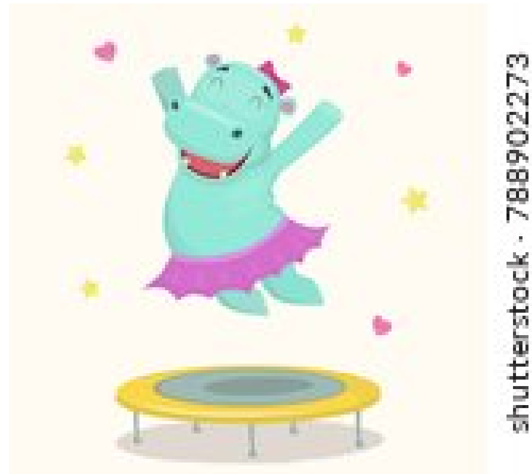
Dancing Hippos

NYU is preparing a new attraction for the graduation ceremonies: A Dancing Hippos Act. The new president of atractions was able to locate a supplier who can provide a large number of dancing hippos. The idea for the act is to have them dancing on a platform that will be raised above the graduates.

The hope is to have as many hippos dancing up on the elevated platform as possible. But the platform has limited weight capcity and we would not want to have new graduates squashed by the falling hippos.



Given the weight capacity `W` of the platforms, the number of dancing hippos available `N`, and the weights of all the hippos, `w_1, w_2, ..., w_N`, we need to figure out the largest number of dancing hippos that can be placed on the platform without endangering the graduates.

Input

First two lines contains `0<=W<=10^8` and `0<=N<=10^5` as described above. The net line contains `N` weights of the dancing hippos, `0<=w_i<=10^8`.

Output

Maximum number of hippos that can be safely placed on the platform.

Example 1

```
Input
50
4
76 56 64 69


Output
0
```

Example 2

```
Input
150
7
45 32 78 56 22 63 37

Output
4
```

Example 3

```
Input
10000
10
87 21 15 43 60 61 71 57 79 20

Output
10
```

Sia's Box

Sia has a mysterious box and she wants to play a game with you. The box supports two types of operations:

- `1 x`: Put the number x into the box.
- `2`: Take out a number from the box.

Sia will give you a sequence of operations and the results of the above `2` operations. Your task is to determine what is really hidden in the box: a stack, a queue, a max priority queue or something else.

Input

The first line of the input contains a single integer N (0 <= N <= 1000), indicating the number of operations. Each of the next N lines contains a single operation described above. For operation `2`, there is an additional number x indicating the result of that operation. The value of x in both operations satisfies 1 <= x <= 100.

Output

You should print one of the following answers on a line by itself:

- `stack` if it is certain that the box is a stack.
- `queue` if it is certain that the box is a queue.
- `priority queue` if it is certain that the box is a max priority queue.
- `impossible` if it is certain that the box cannot be any of those three data structures.
- `not sure` if the box could possibly be more than one of those three data structures.

Example 1

```
Input:
6
1 1
1 2
1 3
2 1
2 2
2 3

Output:
queue
```

Example 2

```
Input:
6
1 1
1 2
1 3
2 3
2 2
2 1

Output:
not sure
```

Example 3

```
Input:
2
1 1
2 2

Output:
impossible
```

Example 4

```
Input:
7
1 84
1 36
1 61
1 4
2 4
1 61
2 61

Output:
stack
```

Example 5

```
Input:
7
1 2
1 5
1 1
1 3
2 5
1 4
2 4

Output:
priority queue
```

Safe Steps

Percy is going to Olympus to return the stolen lightening bolt. To his surprise, each of the steps that lead to Zeus's palace are marked with N bits incrementally starting from 0 to 2^N, 1 <= N <= 16. Now, to reach the top safely Percy can only use the steps that have exactly H bits set (he has to step over any other steps), 1 <= H <= N <= 16. Help Percy find the sequence of all the steps on which he can step or else Poseidon and Zeus will destroy the world of mortals if the lightening bolt is not returned.

Input

The input consists of N and H separated by white space on a single line.

Output

Print the numbers of steps one per line from (numerically) smallest to largest. The output should end with a newline.

Example 1

```
Input:

4 1

Output:

0001
0010
0100
1000
```

Example 2

```
Input:

4 2

Output:

0011
0101
0110
1001
1010
1100
```

Pandora's Underwater World

You are in a 3D underwater world of Pandora's eastern sea. This magical world could be represented as a three dimensional arrangment of air and water pockets. You are still unable to breath under water, so you need to find your way through this world using only the air pockets. The air pockets are connected in the typical 3D fashion to its six adjacent neightbors. You can move from one air pocket to another if the two pockets are connected and they are both air pockets. Given your current location and location of the exit to the surface, you need to figure out how to get back to the surface.

Input

The first line of the input contains three integers L, R and C (1 <= L,R,C <= 30), indicating the number of layers, rows and columns of the underwater maze Then there follows L *rectangular areas* of R lines each containing C characters. Each character describes one pocket, where `W` indicates a water pocket and `A` indicates an air pocket. Your current position is indicated by `S` and the exit by `E`. `S` and `E` never *overlap*, they are always at distinct locations and there is always only one of each.

There will be a single blank line after each level.

Output

Print one line `Reached the surface in X minute(s).` where `X` is the minimum time needed to reach the surface (assuming moving from one air pocket to the next takes always one minute), or `Staying forever!` if it is impossible to get to the surface. There should be a new line character at the end of the output statement.

Example 1

```
Input:
3 4 5
SAAAA
AWWWA
AWWAA
WWWAW

WWWWW
WWWWW
WWAWW
WWAAA

WWWWW
WWWWW
WAWWW
WWWWE


Output:
Reached the surface  in 11 minute(s).
```

Example 2

```
Input:
1 3 3
SWW
WEW
WWW


Output:
Staying forever!
```

Example 3

```
3 3 3
WWW
WSW
WWW

WWW
WAW
WWW

WWW
WEW
WWW

Reached the surface  in 2 minute(s).
```

Airline Boarding

After years of hearing complains from its customers about the boarding process, the Best Ever Airline (BEA) decided to completely change it.

The new boarding process will go as follows:

- When a passanger arrives at the check-in desk, they are issued a boarding number.
- When it is time to board, all passangers line up.
- The boarding attendants admit passangers one at a time according to the following rules
  - B is the boarding number of the passanger at the front of the line
  - if any passanger in the line has a boarding number higher than B, then the current front-of-the-line passanger is sent to the back of the line
  - if there is no passanger in the line with a boarding number higher than B, then the current front-of-the-line passanger is allowed to board the plane

You know your own position in line and your own boarding number. You also know the boarding numbers of all the people along the line at the start of the process. Figure out how many passangers board the plane ahead of you.

Input

The first line of input contains two numbers:

- N is the total number of passangers in line, `1 <= N <= 100`
- M is your position in the line, `0 <= M < N`

The second line of input contains N values that represent the boarding numbers of all the passangers in the line (including your own). The boarding numbers range from 1 to 9 (inclusive).

Output

The output is **one line** with a single number: the number of passangers on the plane once you finally board it (do not count yourself).

Example 0

```
Input:
2 1
8 3

Output:
1
```

Your boarding number is 3. The only passanger on the line in front of you has a larger number, so they board ahead of you.

Example 1

```
Input:
2 1
3 8

Output:
0
```

Your boarding number is 8. The only passanger on the line in front of you has a smaller number, so they go back to the end of the line and you board first.

Example 2

```
Input:
5 3
4 5 4 5 4

Output:
1
```

Your boarding number is 5. The first passanger with boarding number 4 goes to the back of the line: 5 4 5* 4 4 (where * indicates your place). The next passanger has boarding number 5 and boards; resulting line: 4 5* 4 4. The next passanger goes to the back of the line; resulting line: 5* 4 4 4. Then you get to board and there is only one passanger on the plane.