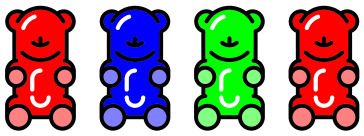
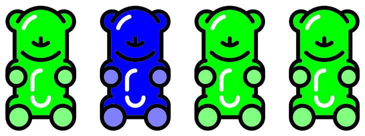
Gummy Bears

Mouse Stofl loves gummy bears, and he loves to eats them in a special way. A pack of gummy bears has n gummy bears. Each gummy bear has some flavour, given as an integer.

Stofl wants to eat the pack by repeatedly taking mouthfuls of k gummybears (where it is guaranteed that k divides n). A mouthful of gummy bears is *yummy* if there is a majority flavour, i.e. there is some flavour of gummy bears that occurs in strictly more than $k/2$ gummy bears of that mouthful. If there is no majority flavour, Stofl finds the mouthful to be *yucky*.

For example, if we have $k = 4$, a mouthful must contain strictly more than $k/2 = 2$, i.e. at least 3 flavours of the same kind.


Yuck! There is no majority flavour.


Yummy! Green is the majority flavour.

Is it possible for Stofl to eat the gummy bears in such a way that all mouthfuls are yummy? If yes, print a possible way to eat them.

Input

The first line of the input contains the three integers n, k and p – the number of gummy bears n, the size of a mouthful k and p, whether you also have to print how to eat it (if $p = 0$ you don't have to print it, if $p = 1$ then you do).

The second line contains n integers f_i – the flavours of the gummy bears.

Output

If it is possible to eat the gummy bears such that all mouthfuls are tasty, print "`Yummy!`" on the first line, else print "`Yuck!`".

If $p = 1$ and your first line was "`Yummy!`", you should also print out a way how to eat them. For that print n/k lines, containing k numbers each: the i-th of those lines should contain the flavours you want to put on the i-th mouthful. You can use each flavour as many times as it occurs in the input.

For $p = 0$, all lines except for the first one are ignored.

If there are multiple possibilities, you may print any of them.

Limits

There are eight subtasks. In all subtasks, $1 \leq k \leq n$.

For odd subtasks (1, 3, 5 and 7), we have $p = 0$.
For even subtasks (2, 4, 6 and 8), we have $0 \leq p \leq 1$.
In the samples, $p = 1$. If you plan to solve only the subtasks with $p = 0$, you can still see in the verdict whether the first line was correct.

- Subtasks 1 and 2, worth 10 points each: $n \leq 100\,000$, $0 \leq f_i < n/k$ and each flavour occurs exactly k times. In other words, the values f_i contain exactly k times 0, k times 1, etc., and k times $n/k - 1$.
- Subtasks 3 and 4, worth 10 points each: $n \leq 100\,000$ and $0 \leq f_i < 2$, i.e. there are at most 2 different flavours.
- Subtasks 5 and 6, worth 10 points each: $n \leq 1000$ and $0 \leq f_i < n$.

- Subtasks 7 and 8, worth 20 points each: $n \leq 100\,000$ and $0 \leq f_i < 10^9$.

Examples

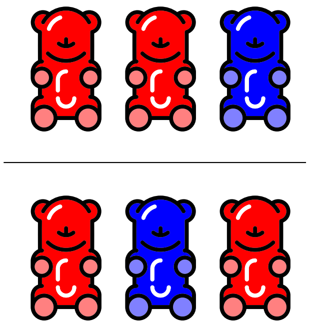
Input

```
6 3 1
1 0 0 0 0 1
```

Output

```
Yummy!
0 0 1
0 1 0
```

This input would be valid for subtasks 3, 4, 5, 6, 7 and 8, if we disregard the value of p. The solution looks like this (with red = 0 and blue = 1):



Input

```
16 4 1
0 0 0 1 1 1 1 1 1 1 1 1 3 3 4 4
```

Output

```
Yummy!
0 0 0 3
1 1 1 3
1 1 1 4
1 1 1 4
```

This input would be valid for subtasks 5, 6, 7 and 8, disregarding p.

Input

```
12 3 1
0 0 0 0 0 1 2 2 2 3 4 5
```

Output

```
Yuck!
```

This input would be valid for subtasks 5, 6, 7 and 8, disregarding p.

Input

```
4 2 1
0 1 0 0
```

Output

```
Yuck!
```

This input would be valid for subtasks 3, 4, 5, 6, 7 and 8, disregarding p.

Input

```
12 4 1
0 1 2 2 1 1 1 0 2 2 0 0
```

Output

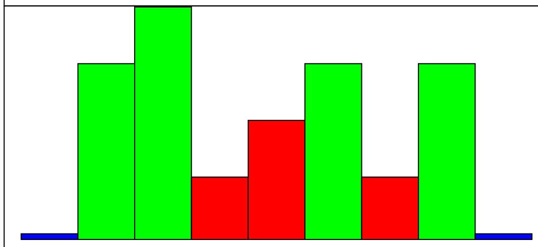
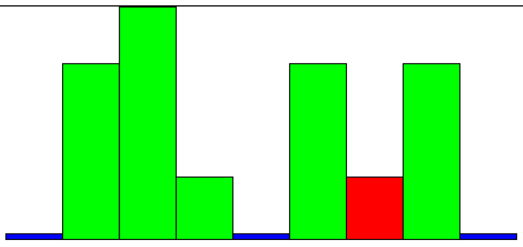
```
Yummy!
0 0 0 0
1 1 1 1
2 2 2 2
```

This input would be valid for subtasks 1, 2, 5, 6, 7 and 8, disregarding p.

Water View

Mouse Stofl is constructing a hiking path between two lakes. The path consists of n segments, where segment i is at height h_i. The two lakes are at height 0, and are positioned before the 0-th and after the $(n-1)$-th segment, respectively. Stofl wants to make the hike as beautiful as possible. A segment i of the hike is *beautiful* if you can see a lake, i.e. there is a lake such that there is no segment higher than h_i between segment i and that lake. The beauty of the hike is the number of beautiful segments it contains. Stofl may replace exactly one segment with an additional lake, also at height 0. That lake also counts as a beautiful segment. (Since the hike may not have any gaps, Stofl builds a bridge across the lake, from which you can obviously see the lake.)

Below is a visualization of example 3. The beautiful segments are marked green.

Before	**After**
	
This is the situation before building the additional lake. Three of the segments don't have any water view, therefore it has beauty 4.	After building a new lake in the middle, all but one segment become beautiful and the new beauty is 6.

Input

The first line of the input contains an integer n, the number of segments. The next line contains n integers h_i, the heights of the segments.

Output

Print a single integer, the maximal number of beautiful segments Stofl can achieve.

Limits

In all testcases $1 \le h_i \le 10^9$ and $1 \le n \le 500\,000$.

- In the first test group, worth 30 points, $1 \le n \le 500$.
- In the second test group, worth 30 points, $1 \le n \le 10\,000$.
- In the third test group, worth 40 points, there are no further restrictions.

Examples

Input	Output
3 2 1 3	3

Stofl may replace any of the three segments with a lake in order to make all segments beautiful.

Input	Output
7 3 4 1 2 1 3 2	7

The segment with height 2 is the only segment Stofl can convert in order to make all segments beautiful.

Input	Output
7 3 4 1 2 3 1 3	6

Stofl cannot make all segments beautiful. If he converts the segment with height 2, six segments will be beautiful. This example is shown in the visualization above.

Sliding mouse

Mouse Daniel is standing in the middle of an ice field. The ice field is a rectangular grid of size $n \times m$, consisting of three types of tiles:

- Wall tiles "`#`": Mouse Daniel can't move onto these tiles.
- Ice tiles "`+`": When moving onto such a tile, mouse Daniel slides over it, so he'll move off the tile in the same direction (unless there is a wall in that direction, in which case he comes to a stop).
- Dirt tiles "`.`": When moving onto such a tile, mouse Daniel stops. He may move off the tile in any of the 4 directions.

There are two special dirt tiles:

- Start tile "`d`": Mouse Daniel starts here.
- Goal tile "`g`": Mouse Daniel wins if he reaches this tile.

Mouse Daniel moves at a speed of one tile per second. Can you help him reach the goal as soon as possible?

Input

The first line contains two integers n and m. After that, m lines follow, each containing n characters. Each character is one of `#`, `+`, `.`, `d`, `g`. `d` and `g` appear exactly once. All border tiles are `#`.

Output

Output the number of seconds which mouse Daniel needs to reach the goal, or "`IMPOSSIBLE`" if the goal cannot be reached.

Limits

There are 4 test groups, each of which is worth 25 points.

- In test group 1, we have $n, m \leq 100$ and there are no ice tiles.
- In test group 2, we have $n, m \leq 100$.
- In test group 3, we have $n, m \leq 1000$.
- In test group 4, we have $n, m \leq 2000$.

Examples

Input	Output

`8 5`
`########`
`#.++g#.#`
`#.+++.d#`
`#..+..+#`
`########`
	`5`

Daniel can't walk up and then left, because there is a wall in the way. If he walks two steps left, he would slide across the ice up to the left wall, and it would take 9 seconds to reach the goal. A faster way is to first walk down (he will be stopped by the wall), two steps left and then slide across the ice to the goal.

Input	Output

`5 3`
`#####`
`#d#g#`
`#####`
	`IMPOSSIBLE`

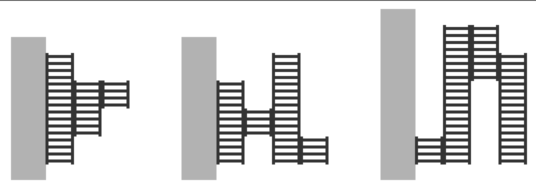
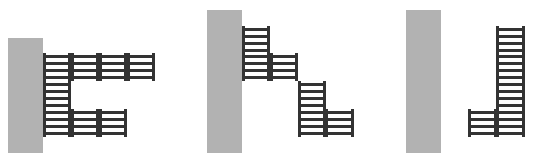
Unfortunately, the goal tile is behind a wall, so mouse Daniel can't reach it.

Ladder Balcony

Mouse Binna has just finished building a huge tower and now she wants to build a balcony for it. She doesn't just want a regular boring balcony, no, the balcony should be made out of ladders. This would allow her to climb up and down while enjoying the sunset.

If we picture the tower on the left and the sunset in the front-right, then the ladders may be set up as follows: The leftmost ladder has to be bolted to the tower and every other ladder has to be bolted to the ladder directly to the left of it. In order for two ladders to be bolted together properly, then need to share an edge, not just a corner.

Additionally, there can never be two ladders above each other with a gap between them, as Mouse Binna could fall down and break her legs if she doesn't see the gap.

Valid	**Invalid**
	

The tower is H mousemeters high and mouse Binna estimates that the tower can support ladders up to W mousemeters out. She has already bought a ladder that is N mousemeters long and 1 mousemeter wide. She may cut this ladder into various pieces, but for aesthetic reasons, all those pieces have to be an integer number of mousemeters long. She will then bolt the pieces together to build her balcony.

Using a drone, Mouse Binna has already mapped out how much sunlight the $H \times W$ area in front of her tower gets. She would like to maximize the total number of sunlight that her balcony gets. Can you help her figure this out?

Input

The first line of the input contains three integers H, W and N – the height of the tower, the maximum distance the balcony may extend outwards and the number of mousemeters of ladder at mouse Binna's disposal. The next H lines each contain W integers $a_{i,j}$ describing the sunlight in front of her tower.

Output

Print a single integer – the maximum total amount of sunlight mouse Binna can have on her balcony.

Limits

In all test cases $1 \leq H, W, N$ and $N \leq H \cdot W$ and $0 \leq a_{i,j} \leq 10^6$.

- In the first subtask (20 points), we have $H, W, N \leq 20$.
- In the second subtask (20 points), we have $H, W, N \leq 40$.
- In the third subtask (20 points), we have $H, W, N \leq 80$.
- In the fourth subtask (20 points), we have $H \leq 100$, $W \leq 80$ and $N \leq 100$.
- In the fifth subtask (10 points), we have $H \leq 120$, $W \leq 50$ and $N \leq 120$.
- In the sixth subtask (10 points), we have $H \leq 130$, $W \leq 30$ and $N \leq 130$.

Examples

Input	Output

`2 3 3`
`1 2 3`
`7 8 9`
	`24`

Input	Output

`7 6 10`
`2 6 1 1 2 9`
`6 2 9 1 9 3`
`7 5 6 8 2 6`
`5 6 5 4 5 4`
`6 7 3 6 3 8`
`4 1 4 5 2 1`
`3 2 5 4 6 3`
	`65`