Codeforces Beta Round 87 (Div. 2 Only)

A. Tram

2 seconds, 256 megabytes

Linear Kingdom has exactly one tram line. It has n stops, numbered from 1 to n in the order of tram's movement. At the i-th stop a_i passengers exit the tram, while b_i passengers enter it. The tram is empty before it arrives at the first stop. Also, when the tram arrives at the last stop, all passengers exit so that it becomes empty.

Your task is to calculate the tram's minimum capacity such that the number of people inside the tram at any time never exceeds this capacity. Note that at each stop all exiting passengers exit **before** any entering passenger enters the tram.

Input

The first line contains a single number n $(2 \le n \le 1000)$ — the number of the tram's stops.

Then n lines follow, each contains two integers a_i and b_i $(0 \le a_i, b_i \le 1000)$ — the number of passengers that exits the tram at the i-th stop, and the number of passengers that enter the tram at the i-th stop. The stops are given from the first to the last stop in the order of tram's movement.

- The number of people who exit at a given stop does not exceed the total number of people in the tram immediately before it arrives at the stop. More formally, $\forall i\ (1 \le i \le n): \sum_{j=1}^{i-1} b_j - \sum_{j=1}^{i-1} a_j \ge a_i$. This particularly means that $a_1 = 0$.
- At the last stop, **all** the passengers exit the tram and it becomes empty. More formally, $\sum_{j=1}^{n-1} b_j - \sum_{j=1}^{n-1} a_j = a_n$.
- No passenger will enter the train at the last stop. That is, $b_n = 0$.

Output

Print a single integer denoting the minimum possible capacity of the tram (0 is allowed).

input
```
4
0 3
2 5
4 2
4 0
```
output
```
6
```

For the first example, a capacity of 6 is sufficient:

- At the first stop, the number of passengers inside the tram before arriving is 0. Then, 3 passengers enter the tram, and the number of passengers inside the tram becomes 3.
- At the second stop, 2 passengers exit the tram (1 passenger remains inside). Then, 5 passengers enter the tram. There are 6 passengers inside the tram now.
- At the third stop, 4 passengers exit the tram (2 passengers remain inside). Then, 2 passengers enter the tram. There are 4 passengers inside the tram now.
- Finally, all the remaining passengers inside the tram exit the tram at the last stop. There are no passenger inside the tram now, which is in line with the constraints.

Since the number of passengers inside the tram never exceeds 6, a capacity of 6 is sufficient. Furthermore it is not possible for the tram to have a capacity less than 6. Hence, 6 is the correct answer.

B. Little Pigs and Wolves

2 seconds, 256 megabytes

Once upon a time there were several little pigs and several wolves on a two-dimensional grid of size $n \times m$. Each cell in this grid was either empty, containing one little pig, or containing one wolf.

A little pig and a wolf are adjacent if the cells that they are located at share a side. The little pigs are afraid of wolves, so there will be at most one wolf adjacent to each little pig. But each wolf may be adjacent to any number of little pigs.

They have been living peacefully for several years. But today the wolves got hungry. One by one, each wolf will choose one of the little pigs adjacent to it (if any), and eats the poor little pig. This process is not repeated. That is, each wolf will get to eat at most one little pig. Once a little pig gets eaten, it disappears and cannot be eaten by any other wolf.

What is the maximum number of little pigs that may be eaten by the wolves?

Input

The first line contains integers n and m $(1 \le n, m \le 10)$ which denotes the number of rows and columns in our two-dimensional grid, respectively. Then follow n lines containing m characters each — that is the grid description. "`.`" means that this cell is empty. "`P`" means that this cell contains a little pig. "`W`" means that this cell contains a wolf.

It is guaranteed that there will be at most one wolf adjacent to any little pig.

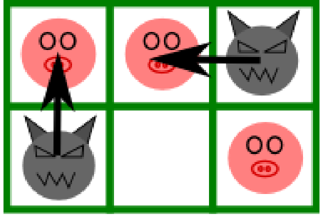
Output

Print a single number — the maximal number of little pigs that may be eaten by the wolves.

input
```
2 3
PPW
W.P
```
output
```
2
```

input
```
3 3
P.W
.P.
W.P
```
output
```
0
```

In the first example, one possible scenario in which two little pigs get eaten by the wolves is as follows.



C. Party

3 seconds, 256 megabytes

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.

input
```
5
-1
1
2
1
-1
```
output
```
3
```

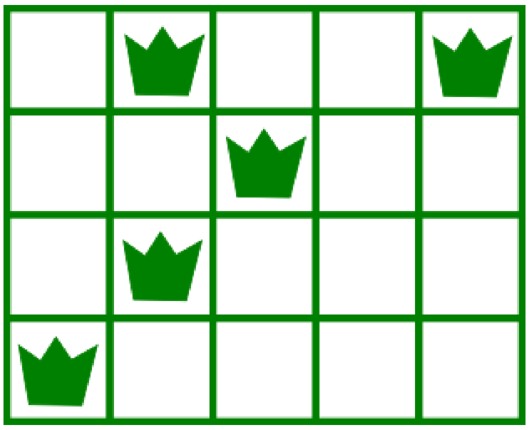
For the first example, three groups are sufficient, for example:

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

D. Lawnmower

2 seconds, 256 megabytes

You have a garden consisting entirely of grass and weeds. Your garden is described by an $n \times m$ grid, with rows numbered 1 to n from top to bottom, and columns 1 to m from left to right. Each cell is identified by a pair (r, c) which means that the cell is located at row r and column c. Each cell may contain either grass or weeds. For example, a 4×5 garden may look as follows (empty cells denote grass):

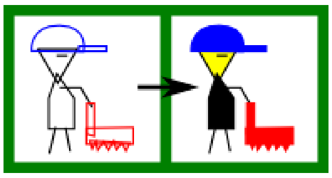


You have a land-mower with you to mow all the weeds. Initially, you are standing with your lawnmower at the top-left corner of the garden. That is, at cell $(1, 1)$. At any moment of time you are facing a certain direction — either left or right. And initially, you face right.

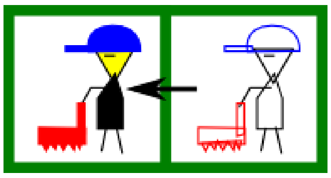
In one move you can do either one of these:

1) Move one cell in the direction that you are facing.

- if you are facing right: move from cell (r, c) to cell $(r, c + 1)$

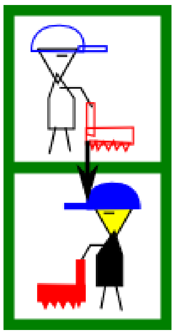


- if you are facing left: move from cell (r, c) to cell $(r, c - 1)$

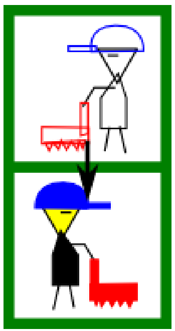


2) Move one cell down (that is, from cell (r, c) to cell $(r + 1, c)$), and change your direction to the opposite one.

- if you were facing right previously, you will face left



- if you were facing left previously, you will face right



You are not allowed to leave the garden. Weeds will be mowed if you and your lawnmower are standing at the cell containing the weeds (your direction doesn't matter). This action isn't counted as a move.

What is the minimum number of moves required to mow all the weeds?

Input

The first line contains two integers n and m $(1 \le n, m \le 150)$ — the number of rows and columns respectively. Then follow n lines containing m characters each — the content of the grid. "G" means that this cell contains grass. "W" means that this cell contains weeds.

It is guaranteed that the top-left corner of the grid will contain grass.

Output

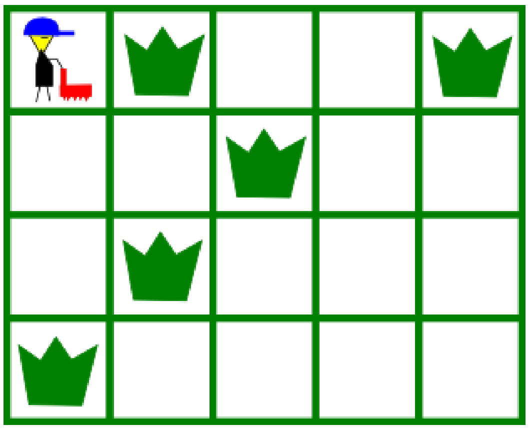
Print a single number — the minimum number of moves required to mow all the weeds.

input
```
4 5
GWGGW
GGWGG
GWGGG
WGGGG
```
output
```
11
```

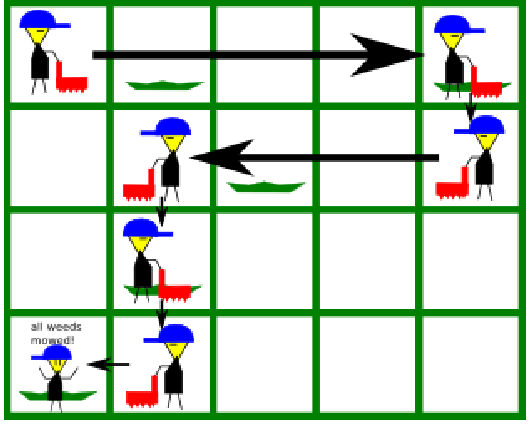
input
```
3 3
GWW
WWW
WWG
```
output
```
7
```

input
```
1 1
G
```
output
```
0
```

For the first example, this is the picture of the initial state of the grid:



A possible solution is by mowing the weeds as illustrated below:

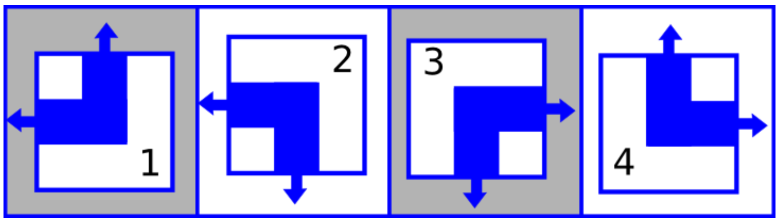


E. Plumber

3 seconds, 256 megabytes

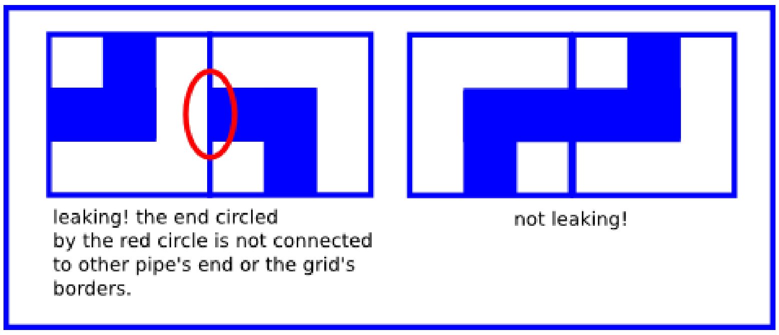
Little John aspires to become a plumber! Today he has drawn a grid consisting of n rows and m columns, consisting of $n \times m$ square cells.

In each cell he will draw a pipe segment. He can only draw four types of segments numbered from 1 to 4, illustrated as follows:



Each pipe segment has two ends, illustrated by the arrows in the picture above. For example, segment 1 has ends at top and left side of it.

Little John considers the piping system to be leaking if there is at least one pipe segment inside the grid whose end is not connected to another pipe's end or to the border of the grid. The image below shows an example of leaking and non-leaking systems of size 1×2.


leaking! the end circled by the red circle is not connected to other pipe's end or the grid's borders.
not leaking!

Now, you will be given the grid that has been partially filled by Little John. Each cell will either contain one of the four segments above, or be empty. Find the number of possible different non-leaking final systems after Little John finishes filling **all** of the empty cells with pipe segments. Print this number modulo 1000003 $(10^6 + 3)$.

Note that rotations or flipping of the grid are not allowed and so two configurations that are identical only when one of them has been rotated or flipped either horizontally or vertically are considered two different configurations.

Input

The first line will contain two single-space separated integers n and m $(1 \le n, m, n \cdot m \le 5 \cdot 10^5)$ — the number of rows and columns respectively. Then n lines follow, each contains exactly m characters — the description of the grid. Each character describes a cell and is either one of these:

- "1" - "4" — a pipe segment of one of four types as described above
- "." — an empty cell

Output

Print a single integer denoting the number of possible final non-leaking pipe systems modulo 1000003 ($10^6 + 3$). If there are no such configurations, print 0.

input

```
2 2
13
..
```

output

```
2
```

input

```
3 1
1
4
.
```

output

```
0
```

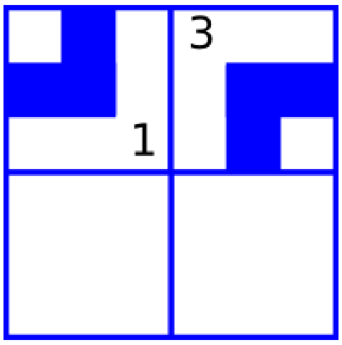
input

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

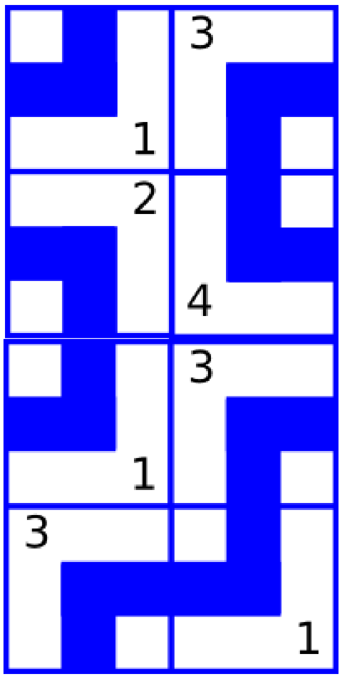
output

```
1
```

For the first example, the initial configuration of the grid is as follows.



The only two possible final non-leaking pipe configurations are as follows:



For the second example, the initial grid is already leaking, so there will be no final grid that is non-leaking.

For the final example, there's only one possible non-leaking final grid as follows.