## Problem 2 – 3D Labyrinth

You are given a 3D Labyrinth, divided into **cubes**. The labyrinth consists of **levels** (floors). Essentially, each **level is a horizontal 2D matrix of cubes**.

You can move in the 3D labyrinth just as you would in an apartment building – you **can move to each of your neighboring cubes on the current level**. Two **cubes** on a floor are **neighbors if they share a common wall** (so each cube has a front, back, left and right neighbor, except for the cubes on the edges and sides, which obviously don't have all 4 neighbors).

In some cases, **when there's a ladder or stairway**, you can also **move from one floor to the other**, that is, the **cube** **directly above or below** you becomes also one of your neighbors.

The number of levels in the labyrinth is exactly **L**. The number of columns for each level is **C** and the number of rows in each level is **R**.

Each cube in the labyrinth can be defined by its position – the level, the row and the column on which the cube is. So if a cube is defined as being on position **(X, Y, Z)**, that means it is on the **Xth level**, the **Yth row** and the **Zth column**. Levels, rows and columns are **numbered beginning from 0**. For example, the corner cubes of the bottom level are: (0,0,0), (0,0,C-1), (0,R-1,C-1), (0,R-1,0) – in clockwise order, if you imagine the first level as a matrix.

**Some of the cubes** in the labyrinth **are impassable**. They are filled with stone and you cannot move into or through them.

**Other cubes** in the labyrinth **are ladders** – some of them allow you to **move to the upper level**, some allow you to **move to the lower level**. Every **ladder is one-directional** – it only allows you to **travel either up or down** – which means that if you went up one ladder, you cannot go down the same ladder.

Find **minimum number of moves** that are required to **escape the labyrinth**,given a **starting location** and the cubes in the labyrinth. To escape the labyrinth you need to be **on top of the labyrinth** (above highest level) or **below the lowest level**. For example if the given labyrinth has 6 levels (0 to 5) you need to be on level -1 or level 6 in order to escape it).

### Input

The input data should be read from the console.

On the first line there will be the numbers **X**, **Y** and **Z**, separated by spaces. These numbers represent your **starting location** – X is the starting level, Y is the starting row, Z is the starting column.

On the next line, of the standard input, there will be the numbers **L**, **R** and **C**, separated by whitespaces.

On the following lines, until the end of the input, there will be **L** **descriptions** of **RxC** **matrices**. Each matrix is represented on exactly **R lines**, each containing exactly **C symbols**. The **first matrix** corresponds to the **first level** of the labyrinth; the **second matrix** corresponds to the **second level** of the labyrinth and so on.

Each symbol in a matrix description can be:

* '**.**' (dot) – meaning an empty cube ,'**U**' – meaning a ladder to the upper level (X+1), '**D**' – meaning a ladder to the lower level (X-1), '**#**' – meaning an impassable cube

The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

The output data should be printed on the console.

On the only output line, print the **minimum** **number of moves** needed to escape the labyrinth.

### Constraints

* **L, R and C** will be between 1 and 100, inclusive. **X, Y and Z** will be between 0 and 99, inclusive.
* There will always be way to escape the labyrinth
* The start position will never be an impassible cube ('#')
* There are NO special restrictions on positioning of the labyrinth elements ('#', 'U', 'D', '.')

|  |  |
| --- | --- |
| **Example input** | **Example output** |
| 2 0 2  3 4 3  ...  .#.  .#.  .#D  ...  ...  ##.  D..  ..D  ...  ##.  ... | 16 |

* Allowed working time for your program: 0.3 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |
| --- | --- | --- |
| **Example input** | **Example output** | **Explanation** |
| 0 0 0  2 3 4  **.#U.**  **..#.**  **U...**  **..D.**  **....**  **..U.** | 6  0 0 0  2 3 4  .#U.  ..#.  U...  ..D.  ....  ..U. | **. # U .**  **. . # .**  **U . . .**  **. . D .**  **. . . .**  **. . U .** |