Castle on the Grid ☆

Problem Submissions Leaderboard Editorial

RATE THIS CHALLENGE

☆ ☆ ☆ ☆ ☆

You are given a square grid with some cells open (**.**) and some blocked (**X**). Your playing piece can move along any row or column until it reaches the edge of the grid or a blocked cell. Given a grid, a start and an end position, determine the number of moves it will take to get to the end position.

For example, you are given a grid with sides $n = 3$ described as follows:

```
...
.X.
...
```

Your starting position $(startX, startY) = (0, 0)$ so you start in the top left corner. The ending position is $(goalX, goalY) = (1, 2)$. The path is $(0, 0) \to (0, 2) \to (1, 2)$. It takes 2 moves to get to the goal.

Function Description

Complete the minimumMoves function in the editor. It must print an integer denoting the minimum moves required to get from the starting position to the goal.

minimumMoves has the following parameter(s):

- grid: an array of strings representing the rows of the grid
- startX: an integer
- startY: an integer
- goalX: an integer
- goalY: an integer

Input Format

The first line contains an integer n, the size of the array grid.

Each of the next n lines contains a string of length n.

The last line contains four space-separated integers, $\mathrm{startX}, \mathrm{startY}, \mathrm{goalX}, \mathrm{goalY}$

Constraints

- $1 \leq n \leq 100$
- $0 \leq startX,\ startY,\ goalX,\ goalY < n$

Output Format

Print an integer denoting the minimum number of steps required to move the castle to the goal position.

Sample Input

```
3
.X.
.X.
...
0 0 0 2
```

Sample Output

```
3
```

Explanation

Here is a path that one could follow in order to reach the destination in 3 steps:

$(0,0) \rightarrow (2,0) \rightarrow (2,2) \rightarrow (0,2)$.

Java 8

```java
import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;

public class Solution {

    private static String[][] grid;
    private static int[][] distanceFromStart;

    /**
     * Applying breadth first search to find the path with least turns from start to
     * goal.
     */
    private static int mininumTurns(int startX, int startY, int goalX, int goalY) {
        if (startX == goalX && startY == goalY) {
```

Line: 1 Col: 1

Upload Code as File □ Test against custom input Run Code Submit Code

Contest Calendar | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature