Blocks Board Problem

You are given a rectangular board divided into a grid of cells. Some cells have blocks (#), others are empty (.).

You can add a block into the field:

- Choose a row index.
- Drop a block from the left; it moves right until a block or edge, then falls until another block or the bottom.

Task: Calculate the minimum and maximum number of moves to fill the first column with blocks.

Constraints:

1 <= field.length <= 12

1 <= field[0].length <= 12

Example 1:

```
field = [[".", "#", "#"],
         ["#", ".", "."],
         [".", ".", "."]]
```

Output: [4, 4]

Example 2 (Modified by user):

```
field = [[".", "#", "#"],
         [".", ".", "#"],
```

[".", ".", "."]]

Output: [3, 6]

Java Solution Template

```
public class BlocksBoardProblem {

    // Function to calculate minimum and maximum moves
    public static int[] solution(char[][] field) {
        int n = field.length;       // number of rows
        int m = field[0].length;    // number of columns

        int minMoves = 0;
        int maxMoves = 0;

        // TODO: Implement logic to compute minimum and maximum moves

        return new int[]{minMoves, maxMoves};
    }

    public static void main(String[] args) {
        char[][] field1 = {
            {'.', '#', '#'},
            {'#', '.', '.'},
            {'.', '.', '.'}
        };

        char[][] field2 = {
            {'.', '#', '#'},
            {'.', '.', '#'},
            {'.', '.', '.'}
        };

        int[] result1 = solution(field1);
        int[] result2 = solution(field2);

        System.out.println("Output 1: [" + result1[0] + ", " + result1[1] + "]");
        System.out.println("Output 2: [" + result2[0] + ", " + result2[1] + "]");
    }
}
```