Programming Assessment

You have 60 minutes to complete the exercise below. At the end of the allotted time, please return your code and answers via email attachment or GitHub/GitLab/Bitbucket link. The focus for your code should be on its clarity which can be accomplished via formatting, comments, and docstrings where appropriate. Partial solutions and solution ideas will be accepted. Please note that this is not the only metric that matters during the assessment process. Try to have fun!

You may complete the exercises in the programming language of your choice; the recommendation is to pick the one you are most comfortable with.

Part 1

Consider a discrete grid of size $n \times n$. Here is an example for $n = 5$.

```
   1  2  3  4  5
  +--+--+--+--+--+
1 |  |  |  |  |  |
  +--+--+--+--+--+
2 |  |  |  |  |  |
  +--+--+--+--+--+
3 |  |  |  |  |  |
  +--+--+--+--+--+
4 |  |  |  |  |  |
  +--+--+--+--+--+
5 |  |  |  |  |  |
  +--+--+--+--+--+
```

Write a function `print_grid(n)` to print a grid of size $n \times n$.

Examples:

```
print_grid(1)
    1
   +--+
 1 |  |
   +--+

print_grid(2)
    1  2
   +--+--+
 1 |  |  |
   +--+--+
 2 |  |  |
   +--+--+

print_grid(3)
    1  2  3
   +--+--+--+
 1 |  |  |  |
   +--+--+--+
 2 |  |  |  |
   +--+--+--+
 3 |  |  |  |
   +--+--+--+
```

Part 2

Inside the grid, we may now delete horizontal or vertical borders. For this, we implement three operations as follows.

(1) Initialization:

Function `new_grid(n)`, given n, returns a data structure / representation which is initialized with a complete grid of size n with all borders present.

```
g = new_grid(5)
print_grid(g)

    1  2  3  4  5
   +--+--+--+--+--+
 1 |  |  |  |  |  |
   +--+--+--+--+--+
 2 |  |  |  |  |  |
   +--+--+--+--+--+
 3 |  |  |  |  |  |
   +--+--+--+--+--+
 4 |  |  |  |  |  |
   +--+--+--+--+--+
 5 |  |  |  |  |  |
   +--+--+--+--+--+
```

(2) Vertical:

```
delete_v (g, i, j):
    in grid g, remove border to the right of row i and column j
```

In the example above, `delete_v(g, 2, 3)` yields the following result:

```
   j ->
      1  2  3  4  5
i    +--+--+--+--+--+
|  1 |  |  |  |  |  |
v    +--+--+--+--+--+
   2 |  |  |     |  |
     +--+--+--+--+--+
   3 |  |  |  |  |  |
     +--+--+--+--+--+
   4 |  |  |  |  |  |
     +--+--+--+--+--+
   5 |  |  |  |  |  |
     +--+--+--+--+--+
```

(3) Horizontal:

```
delete_h (g, i, j):
    in grid g, remove border below row i and column j
```

Continuing from above, applying, `delete_h(g, 4, 3)` gives:

```
      j ->
         1  2  3  4  5
i     +--+--+--+--+--+
|   1 |  |  |  |  |  |
v     +--+--+--+--+--+
      2 |  |  |     |  |
        +--+--+--+--+--+
      3 |  |  |  |  |  |
        +--+--+--+--+--+
      4 |  |  |  |  |  |
        +--+--+  +--+--+
      5 |  |  |  |  |  |
        +--+--+--+--+--+
```

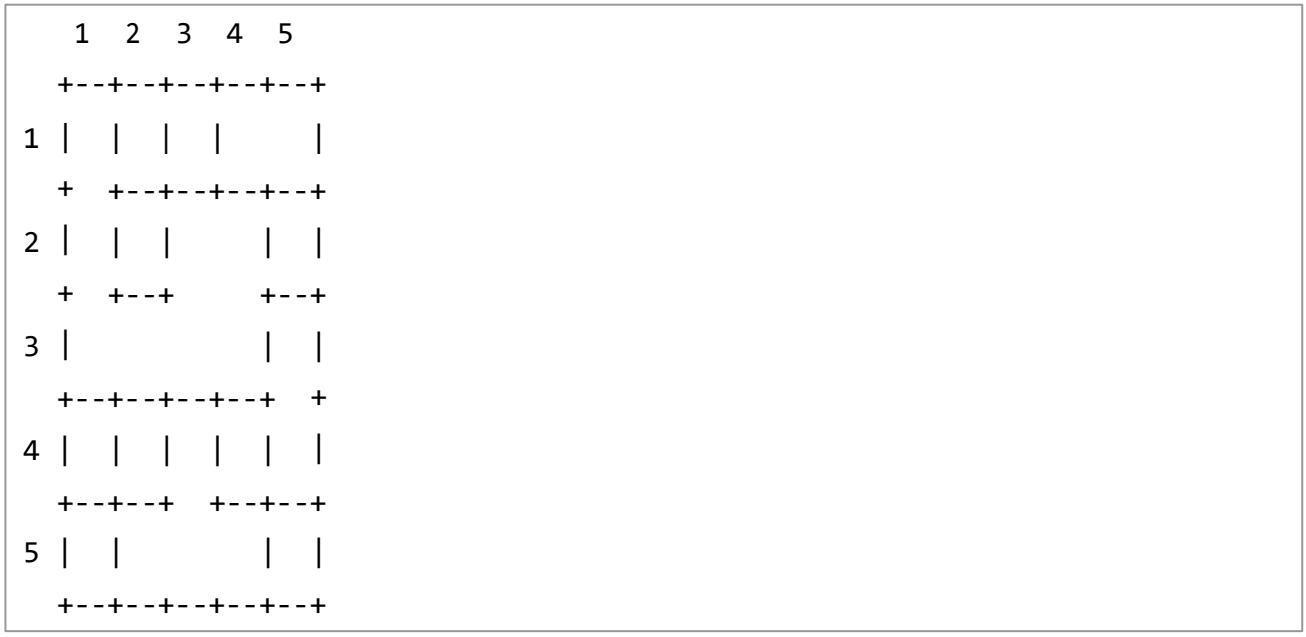
Based on this representation,

a) **implement functions `new_grid`, `delete_h` and `delete_v`, and**

b) **extend the function to print the value of all borders in the format shown above incorporating the deleted borders.**

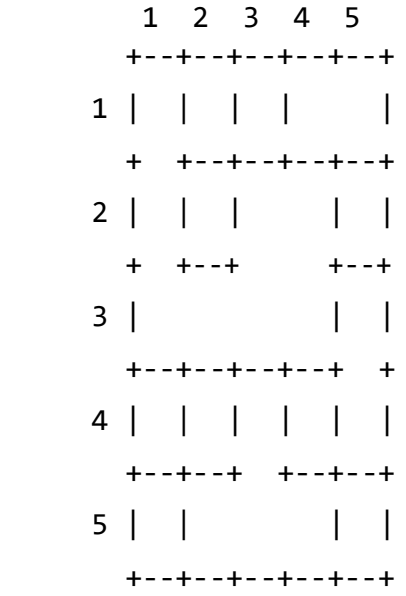
Part 3 (bonus):

Consider a grid where a set of borders have been removed.

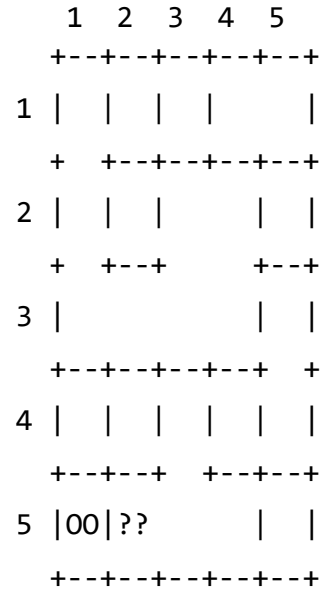


Implement a function path_exists(g, i1, j1, i2, j2), which for a given pair of (row, column) locations a = (i1, j1), b = (i2, j2) returns whether a path between a and b exists. Some illustrations are shown below. Note the function only needs to return True or False, not print the path.

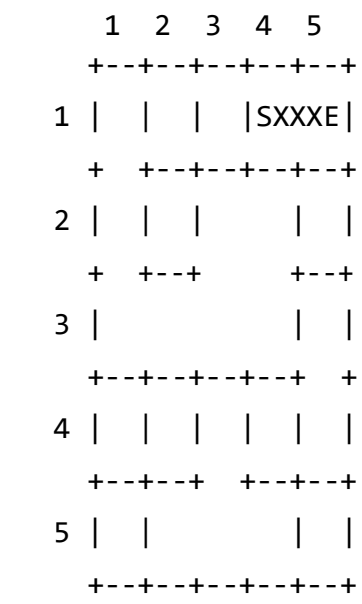
```
print_grid(g)
```



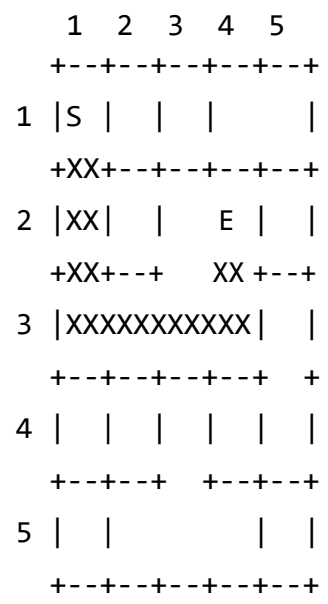
```
path_exists(g, 5, 1, 5, 2) → False
```



path_exists(g, 1, 4, 1, 5) → True



path_exists(g, 1, 1, 2, 4) → True



path_exists(g, 1, 1, 3, 5) → False