LAB-2 - Vacuum Cleaner

Code:

```
class VacuumCleaner:
    def __init__(self, grid):
        self.grid = grid
        self.position = (0, 0)

    def clean(self):
        x, y = self.position
        if self.grid[x][y] == 1:
            print(f"Cleaning position {self.position}")
            self.grid[x][y] = 0
        else:
            print(f"Position {self.position} is already clean")

    def move(self, direction):
        x, y = self.position
        if direction == 'up' and x > 0:
            self.position = (x - 1, y)
        elif direction == 'down' and x < len(self.grid) - 1:
            self.position = (x + 1, y)
        elif direction == 'left' and y > 0:
            self.position = (x, y - 1)
        elif direction == 'right' and y < len(self.grid[0]) - 1:
            self.position = (x, y + 1)
        else:
            print("Move not possible")

    def run(self):
        rows = len(self.grid)
        cols = len(self.grid[0])


        for i in range(rows):
            for j in range(cols):
                self.position = (i, j)
                self.clean()
```

```
        print("Final grid state:")
        for row in self.grid:
            print(row)

def get_dirty_coordinates(rows, cols, num_dirty_cells):
    dirty_cells = set()
    while len(dirty_cells) < num_dirty_cells:
        try:
            coords = input(f"Enter coordinates for dirty cell {len(dirty_cells) + 1} (format: row,col):
")
            x, y = map(int, coords.split(','))
            if 0 <= x < rows and 0 <= y < cols:
                dirty_cells.add((x, y))
            else:
                print("Coordinates are out of bounds. Try again.")
        except ValueError:
            print("Invalid input. Please enter coordinates in the format: row,col")
    return dirty_cells

rows = int(input("Enter the number of rows: "))
cols = int(input("Enter the number of columns: "))
num_dirty_cells = int(input("Enter the number of dirty cells: "))

if num_dirty_cells > rows * cols:
    print("Number of dirty cells exceeds total cells in the grid. Adjusting to maximum.")
    num_dirty_cells = rows * cols

initial_grid = [[0 for _ in range(cols)] for _ in range(rows)]
dirty_coordinates = get_dirty_coordinates(rows, cols, num_dirty_cells)

for x, y in dirty_coordinates:
    initial_grid[x][y] = 1

vacuum = VacuumCleaner(initial_grid)
print("Initial grid state:")
for row in initial_grid:
    print(row)

vacuum.run()
```

Output:

```
Enter the number of rows: 2
Enter the number of columns: 2
Enter the number of dirty cells: 1
Enter coordinates for dirty cell 1 (format: row,col): 0,1
Initial grid state:
[0, 1]
[0, 0]
Position (0, 0) is already clean
Cleaning position (0, 1)
Position (1, 0) is already clean
Position (1, 1) is already clean
Final grid state:
[0, 0]
[0, 0]
>>>
```