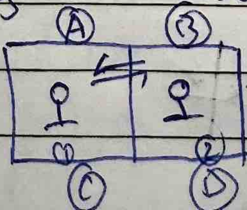
LAB-2 - Vaccum Cleaner

Algorithm:

1. Initialize
   - create 2 2D grids to represent room1 and room2.
   - set initial position to top of room1 [0,0]
2. Percive environmet
   - at each step vaccum cleaner checks if cell is dirty, then cleans it, if not then it moves to another step.
3. Clean
   - after cleaning the dirty cell, change state to clean.
4. Movement
   - Left - right pattern
   - Move right if its not the end.
   - Step down if its the end
5. Check
   - If room 1 is clean move to room2.
6. Repeat
   - repeat the above steps for room2.
7. End
   - vaccume cleaner registers that both rooms are clean.
8. Results
   - Display the results of both rooms.

Rooms



① (A, left),

① (A, left), ② (clean, B)

① (A, left) ② (clean, B) ③ (A right)

Code:-

```
class  vc:
    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("Cleaning position {self.position}")
            self.grid [x][y]=0
        else:
            print(" position {self-position}is already clean)
    def move(self,direction):
        x,y = self.position
        if direction =='up' and x>0:
            self.positon=(x-1,y)
        elif direction=='down'an x<len(self.grid)-1;
            selfpoction=(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 dirty(rows, cols, ndcells):
        dcells=set()
        while len(dcells) < ndcells:
            try
                coords=input("Enter co-ordinates for dirty cell
                        {len(dcells)+1} (format: row,col):")
                x,y=map(int coords.split(','))
                if 0<=x<rows and 0<=y<cols:
                    dcells.add((x,y))
                else:
                    print("Coordinates are out of bounds
                            try again")
            except ValueError:
                print("Invalid input")
        return dcells
    rows=int(input("Enter the no-of rows:"))
    cols=int(input("Enter the no-of columns:"))
    ndcells=int(input("Enter the no-of dirty cells:"))
    if ndcells>rows*cols:
        print("Number of dirty cells exceeds total
                cells. Adjusting to maximum")
        ndcells=rows*cols
    initial_grid=[[0 for _ in range(cols)] for _ in range(rows)]
    dirty_coords = dirty(rows, cols, ndcells).
```

```
for x,y in dirty_coords:
    initial_grid[x][y]=1
~~vaccum = VaccumCleaner(initial_grid)~~
vaccum = vc(initial_grid)
print("initial grid state:")
for row in initial_grid:
    print(row)
vacuum.run()
```

:")

Output:

```
Enter the no.of rows:2
Enter the no.of columns: 2
Enter the no.of dirty cells:1
Enter the coordinate 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]
```

Sneha B
3/10/24

(row)]