Lab-6

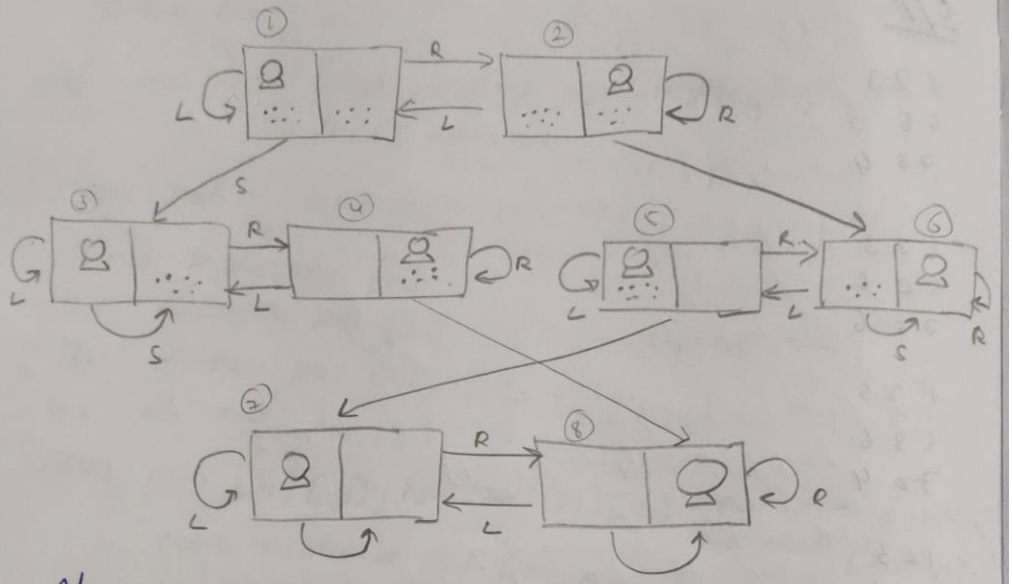
22/12/23

→ Implement the vacuum cleaning agent

$(n.2^n)$ => n = number of room (2)

$2 \cdot 2^2 = 8$ state

22-12-23



Algorithm

1) init (self, room1, room2): —> constructor
   → initialize the cleaning agent with 2 rooms & set the current room

2) clean_room (self): initiates the cleanly process of each room, it call the 'clean_room function' to each room & the it calls the 'move_to_next_function'.

3) def clean_room (self, room):

```
if (room == 'clean'):
    return (clean)
else if (room == 'dirty'):
    initiates the cleaning process
    room == 'clean'
    return 'clean'
```

4) move_to_next:

count = 0:

→ Switch the current room to adjacent room.

if count $n.2^n \leq 2$:

end

else:
move_to_new_room.

Code

```
class VacuumCleaner:
    def __init__(self, initial_location):
        self.location = initial_location

    def move_left(self):
        print("Moving Left")
        self.location = 'A'

    def move_right(self):
        print("Moving right")
        self.location = 'B'

    def suck(self, room):
        print("Sucking dirt in Room{room}")
        return 'clean'

def simulate_cleaning():
    initial_vacuum_location = ip("Enter initial location
        of the vacuum cleaner (A/B):").upper()
    Vacuum = VacuumCleaner(initial_vacuum_location)
    room_A_state = I/P("Enter state for Room A
        (clean/dirty):").lower()
    room_B_state = I/P("Enter state for Room B
        (clean/dirty):").lower()

    rooms = {
        'A': room_A_state,
        'B': room_B_state,
    }

    print("\n initial state:")
    print("Vacuum cleaner is in Room
    {vacuum.location}")
```

```
print(f"Room A: {rooms['A']}")
print(f"Room B: {rooms['B']}\n")

if rooms['A'] == 'clean' and rooms['B'] == 'clean':
    print("Both rooms are already clean. No
          cleaning needed.")
else:
    print("Starting the cleaning process")

    current_room = vacuum.location
    cleaned_room = vacuum.suck(current_room)

    if cleaned_room == 'clean':
        rooms[current_room] = 'clean'

    if current_room == 'A':
        vacuum.move_right()
        current_room = 'B'
    else:
        vacuum.move_left()
        current_room = 'A'

clean_room = vacuum.suck(current_room)
    if cleaned_room == 'clean':
        rooms[current_room] = 'clean'

    print("\nCleaning completed.")
    print("Final state:")
    print(f"Room A: {rooms['A']}")
    print(f"Room B: {rooms['B']}")

simulate_cleaning()
```

Op

Enter initial location of vacum cleaner (A/B): A
Enter state for Room A (clean/dirty): dirty
Enter state for Room B (clean/dirty : dirty

Initial state :

Vacum cleaner is in Room A
Room A: Dirty
Room B : dirty

Starting the cleaning process
Sucking dirt in Room A
moving right
Sucking dirt in Room B

cleaning completed
Finial State :
vacum cleaner is in Room B
Room A : clean
Room B : clean.

Output:

```
0 indicates clean and 1 indicates dirty
Enter Location of VacuumB
Enter status of B0
Enter status of other room1
Vacuum is placed in location B
0
Location B is already clean.
Location A is Dirty.
Moving LEFT to the Location A.
COST for moving LEFT 1
Cost for SUCK 2
Location A has been Cleaned.
GOAL STATE:
{'A': '0', 'B': '0'}
Performance Measurement: 2
```

```
[ ]  Start coding or generate with AI.
```