22/12/2023

5

Lab-6

→ Implement the vacuum cleaner agent.

<u>Algorithm:</u>

=) Vacccumn-cleaning-Agent class:

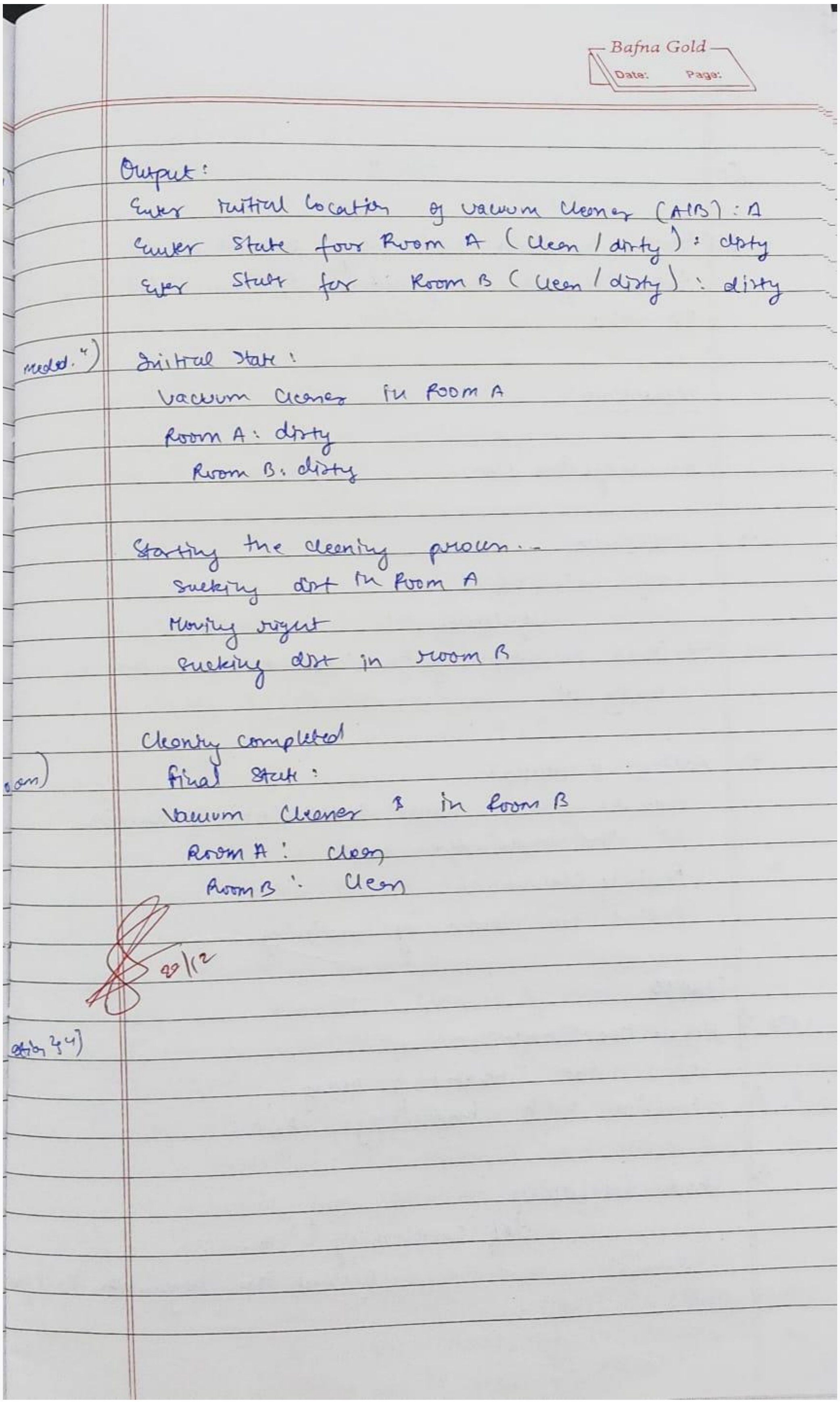
```
1 → __init__(self, room1, room2): → constructor.
      → initialize the cleaning agent with 2 rooms
        and set the current room.

2→ def Clean-rooms(self): Initiates the cleaning process of
      each rooms, it calls the 'clean-room' function
      to each room & then it calls the
      'move_to_next-function'.

3→ def Clean-room(self, room):
        if (room == 'clean'):
            return (clean)
        ~~else~~ elif (room == 'dirty'):

            initiates the cleaning process.
            room == 'clean'
                return (clean).

4→ def move_to_next_room:
        Count = 0:
   → Switches the current room to adjacent room.
     if Count <= 2:
        ~~stop~~
        end
     else:
        Move_to_next_room
```

Bafna Gold
Date: Page:

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}")
        return 'clean'

def simulate_cleaning():
    initial_vacuum_location = input (" Enter initial location
                                     ).upper()

    vacuum = Vacuum Cleaner (initial_vacuum_location)

    room_A_state = input (" Enter state for Room A ").
                                     lower()

    room_A_state = input (" Enter state for Room B: ")
                                     lower()

    rooms = {
            'A': room_A_state;
            'B': room_B state.
    }
```

```
print (" \n Initial state:")
print (" Vacuum cleaner is in Room {vacuum.location}")
print (" Room A: {rooms['A']}")
print (" Room B: {rooms['B']}\n")

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

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

    if cleaned_room == 'A':
            vacuum.move_right ()
            current_room = 'A'

    cleaned_room = ~~vacuum~~ vacuum.suck (current_ro

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

print (" \n ~~Star~~ cleaning completed.")
print (" final state:")
print (" vacuum cleaner is in room {vacuum.loc
print (" Room A: {rooms ['A']}")
print (" Room B: {room ['B']}")

simulate_cleaning ()
```

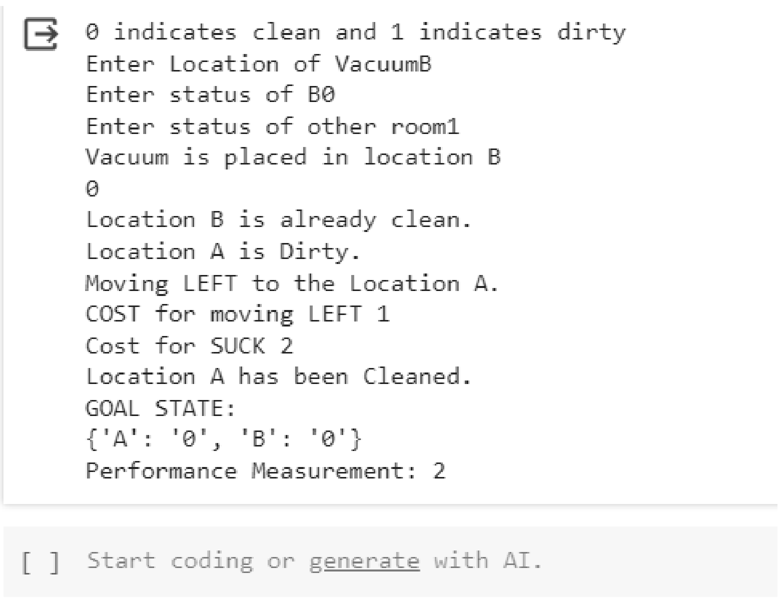


Output:

```
Enter initial location of vacuum cleaner (A/B) : A
Enter State four Room A ( clean / dirty ) : dirty
Enter State for Room B ( clean / dirty ) : dirty

Initial State :
  Vacuum cleaner 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
  final State :
  Vacuum Cleaner is in Room B
    Room A : clean
      Room B : clean
```

[illegible] 22/12

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
```