VACCUM CLEANER

IMPLEMENTATION

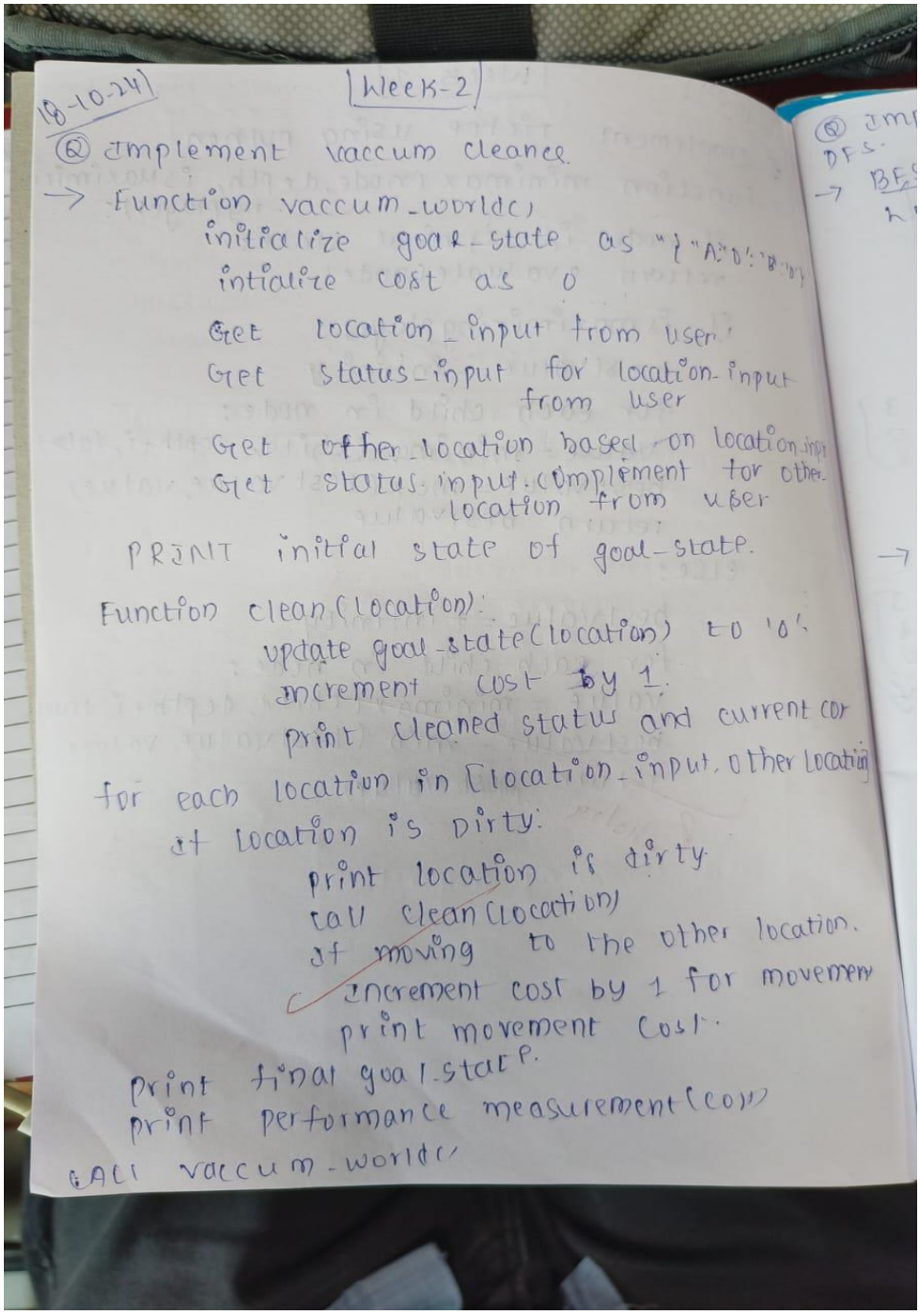
```
print("USN: 1BM22CS324")
print("V DHANUSH REDDY")
class VacuumCleaner:
    def __init__(self):
        # Initialize places A and B as either 'Dirty' or 'Clean'
        self.places = {'A': 'Dirty', 'B': 'Dirty'}
        # Start the vacuum cleaner at place A
        self.current_position = 'A'

    def check(self):
        # Check if the current position is dirty
        if self.places[self.current_position] == 'Dirty':
            print(f"Place {self.current_position} is Dirty.")
            return True
        else:
            print(f"Place {self.current_position} is Clean.")
            return False

    def suck(self):
        # Clean the current position if it's dirty
        if self.check():
            print(f"Cleaning place {self.current_position}.")
            self.places[self.current_position] = 'Clean'
        else:
            print(f"Place {self.current_position} is already clean.")
```

```
    def move(self):
        # Move to the other place
        self.current_position = 'B' if self.current_position == 'A' else 'A'
        print(f"Moving to place {self.current_position}.")

    def start_cleaning(self):
        # Start the cleaning process
        for _ in range(2):  # Loop twice to cover both places
            self.suck()      # Clean the current position if dirty
            self.move()      # Move to the other position

# Create a vacuum cleaner instance
vacuum = VacuumCleaner()

# Start the cleaning process
vacuum.start_cleaning()
```

OUTPUT:

```
USN: 1BM22CS324
V DHANUSH REDDY
Place A is Dirty.
Cleaning place A.
Moving to place B.
Place B is Dirty.
Cleaning place B.
Moving to place A.
```

OBSERVATION:



10-10-24

Week-2

② Implement vaccum cleaner

→ Function vaccum_world()
initialize goal_state as "{"A":'0','B':'0'}"
intialize cost as 0

Get location_input from user
Get status_input for location_input from user
Get other_location based on location_input
Get status_input_complement for other location from user

PRINT initial state of goal_state.

Function clean(location):
update goal_state(location) to '0'
increment cost by 1.
print cleaned status and current cost

for each location in [location_input, other location]
if location is dirty:
print location is dirty
call clean(location)
if moving to the other location.
increment cost by 1 for movement
print movement cost.

print final goal_state.
print performance measurement(cost)

CALL vaccum_world()