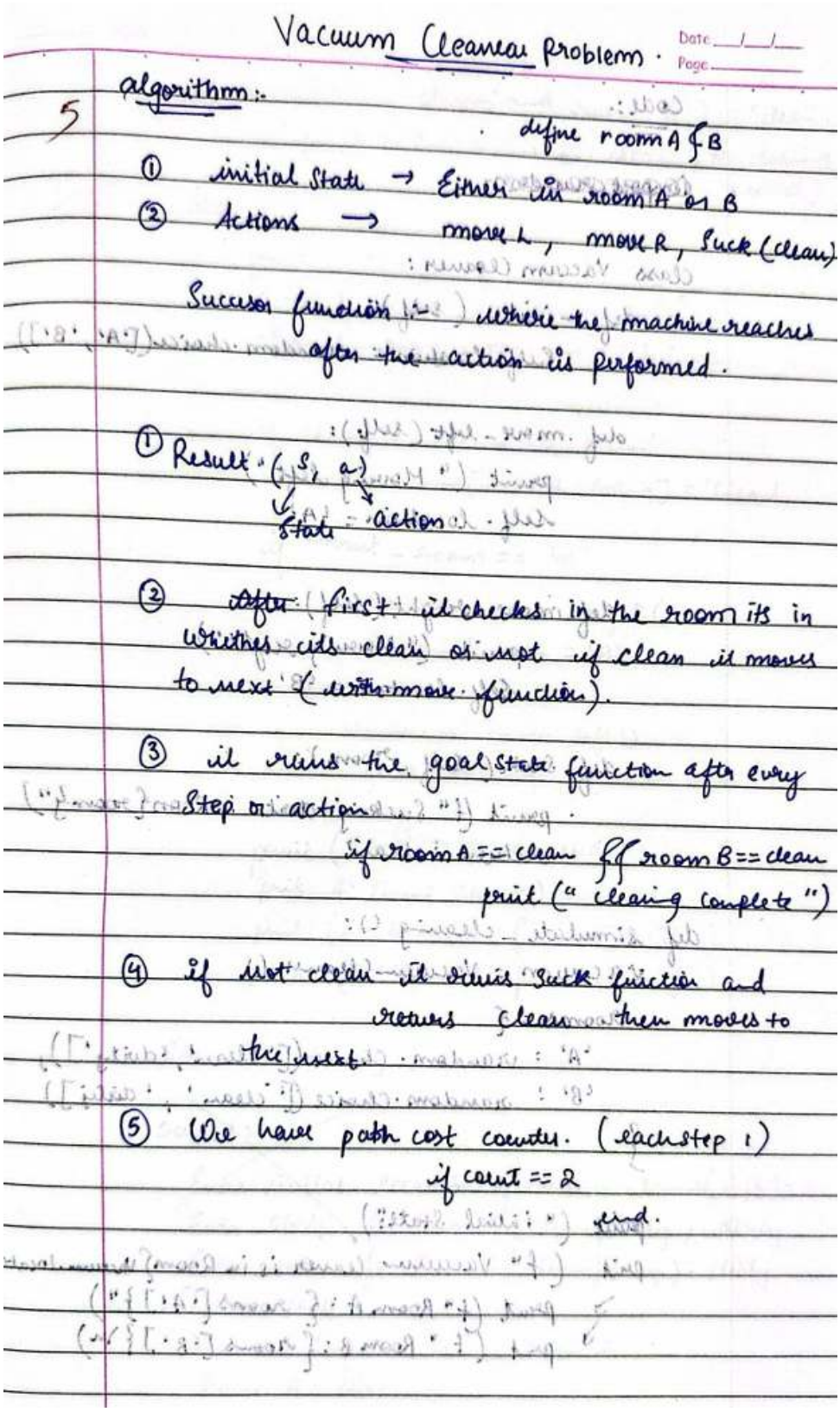
Vacuum cleaner:-
Code:-



Vacuum Cleaner Problem.

Date __/__/__
Page

algorithm:-

- define room A & B

① initial state → Either in room A or B

② Actions → move L, move R, Suck (clean)

Succesor function :- where the machine reaches after the action is performed.

① Result (S, a)
state action

② first it checks in the room its in whether its clean or not if clean it moves to next (move function).

③ it runs the goal state function after every step or action.

if roomA == clean && roomB == clean
print ("cleaning complete")

④ if not clean it runs suck function and returns clean then moves to the next.

⑤ We have path cost counter. (each step 1)

if count == 2
end.

Code:-

```
import random

class VacuumCleaner:
    def __init__(self):
        self.location = random.choice(['A', 'B'])

    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(f"Sucking dirt in Room {room}")
        return 'clean'

def simulate_cleaning():
    vacuum = VacuumCleaner()
    rooms = {
        'A': random.choice(['clean', 'dirty']),
        'B': random.choice(['clean', 'dirty'])
    }

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

```
if room['A']=='clean' and rooms['B']=='clean':
        print("Both rooms are 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'

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

Simulate_cleaning()
```

Output:-

```
Enter initial location of vacuum cleaner(A/B): A
Enter state for Room A (clean/dirty): dirty
Enter State for Room B (clean/dirty): dirty
Initial State:-
Vacuum 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.

Final State:

Vacuum cleaner is in Room B

Room A : clean

Room B : clean

Output:-

main.py

```
        nonlocal cost
        if goal_state[room] == 1:
            print(f"Cleaning Room {room}...")
            goal_state[room] = 0
            cost += 1  # Cost for cleaning
            print(f"Room {room} has been cleaned. Current cost: {cost}")
        else:
            print(f"Room {room} is already clean.")

    # Cleaning logic
    rooms = ['A', 'B', 'C', 'D']
    current_index = rooms.index(location_input)

    # Clean all rooms starting from the initial location
    for i in range(current_index, len(rooms)):
        clean_room(rooms[i])

    # Clean remaining rooms (if the initial location was not 'A')
    for i in range(0, current_index):
        clean_room(rooms[i])

```

input

```
Enter Initial Location of Vacuum (A/B/C/D): B
Enter status of each room (1 - dirty, 0 - clean):
Status of Room A: 1
Status of Room B: 0
Status of Room C: 1
Status of Room D: 1
Initial Location Condition: {'A': 1, 'B': 0, 'C': 1, 'D': 1}
Room B is already clean.
Cleaning Room C...
Room C has been cleaned. Current cost: 1
Cleaning Room D...
Room D has been cleaned. Current cost: 2
Cleaning Room A...
Room A has been cleaned. Current cost: 3
Final State of Rooms: {'A': 0, 'B': 0, 'C': 0, 'D': 0}
Performance Measurement (Total Cost): 7


...Program finished with exit code 0
Press ENTER to exit console.
```