Program 5 : Vacuum Cleaner

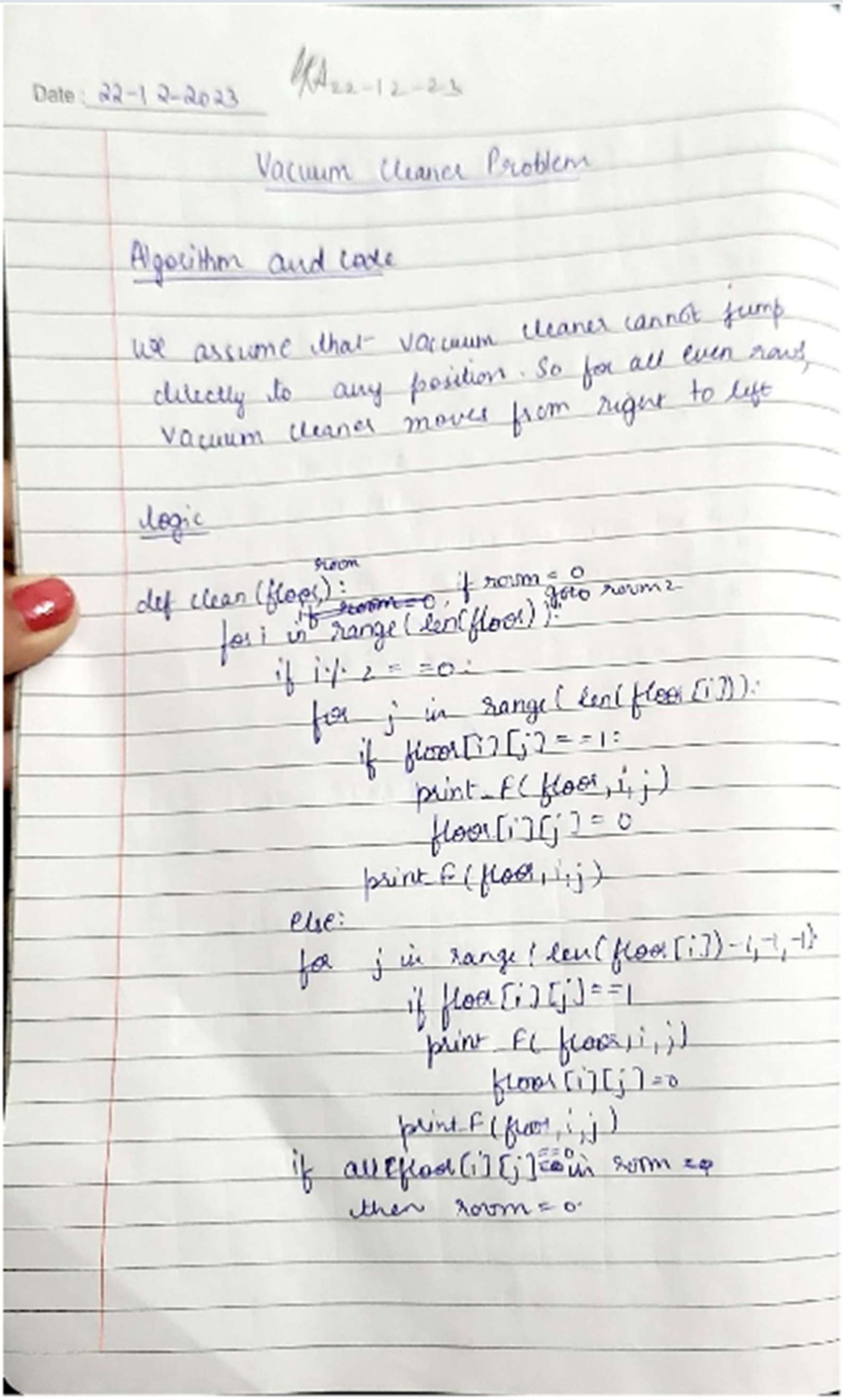
Code:

```
def clean_room(floor, room_row, room_col):
    if floor[room_row][room_col] == 1:
        print(f"Cleaning Room at ({room_row + 1}, {room_col + 1}) (Room was dirty)")
        floor[room_row][room_col] = 0
        print("Room is now clean.")
    else:
        print(f"Room at ({room_row + 1}, {room_col + 1}) is already clean.")

def main():
    rows = 2
    cols = 2
    floor = [[0, 0], [0, 0]]  # Initialize a 2x2 floor with clean rooms

    for i in range(rows):
        for j in range(cols):
            status = int(input(f"Enter clean status for Room at ({i + 1}, {j + 1}) (1 for dirty,
0 for clean): "))
            floor[i][j] = status

    for i in range(rows):
        for j in range(cols):
            clean_room(floor, i, j)

    print("Returning to Room at (1, 1) to check if it has become dirty again:")
    clean_room(floor, 0, 0)  # Checking Room at (1, 1) after cleaning all rooms

if __name__ == "__main__":
    main()
```

Four rooms:

```
def clean_room(room_name, is_dirty):
    if is_dirty:
        print(f"Cleaning {room_name} (Room was dirty)")
        print(f"{room_name} is now clean.")
        return 0  # Updated status after cleaning
    else:
        print(f"{room_name} is already clean.")
        return 0  # Status remains clean

def main():
    rooms = ["Room 1", "Room 2"]
    room_statuses = []
    for room in rooms:
        status = int(input(f"Enter clean status for {room} (1 for dirty, 0 for clean): "))
        room_statuses.append((room, status))
    print(room_statuses)

    for i, (room, status) in enumerate(room_statuses):
        room_statuses[i] = (room,clean_room(room, status)) # Update status after cleaning

    print(f"Returning to {rooms[0]} to check if it has become dirty again:")
    room_statuses[0]=status = (rooms[0],clean_room(rooms[0], room_statuses[0][1])) # Checking
Room 1 after cleaning all rooms

    print(f"{rooms[0]} is {'dirty' if room_statuses[0][1] else 'clean'} after checking.")

if __name__ == "__main__":
    main()
```

Observation:



Date: 22-12-2023 KA 22-12-23

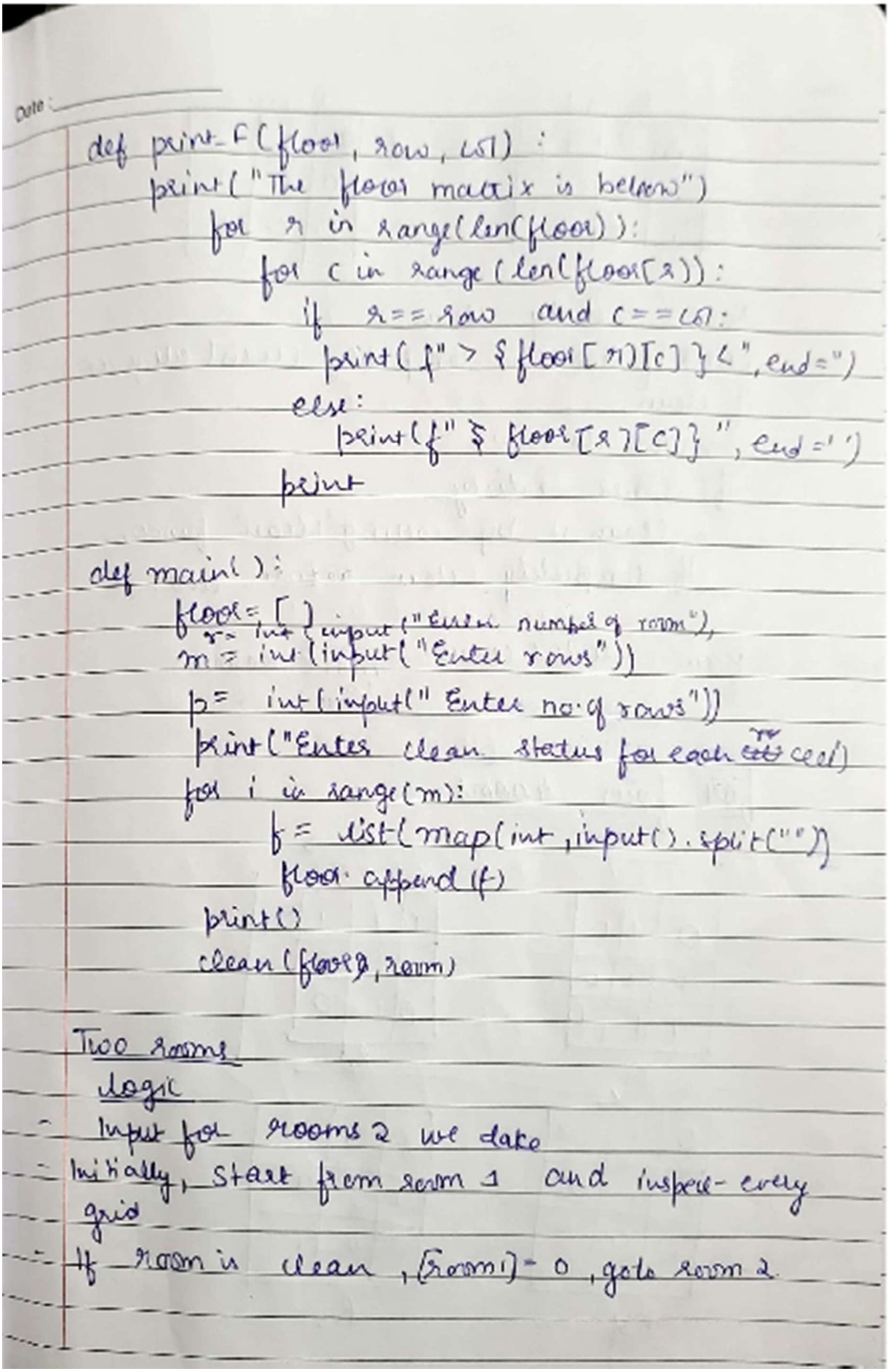
Vacuum Cleaner Problem

Algorithm and code

We assume that vacuum cleaner cannot jump directly to any position. So for all even rows, vacuum cleaner moves from right to left

logic

```
def clean(floors, room):  ~~if room=0~~, if room = 0 goto room2
    for i in range(len(floor)):
        if i%2==0:
            for j in range(len(floor[i])):
                if floor[i][j]==1:
                    print_F(floor,i,j)
                     floor[i][j]=0
                print_F(floor,i,j)
        else:
        for j in range(len(floor[i])-1,-1,-1)
                if floor[i][j]==1
                  print F(floor,i,j)
                        floor[i][j]=0
                print_F(floor,i,j)
        if all(floor[i][j]==0 in room =φ
            then room=0.
```



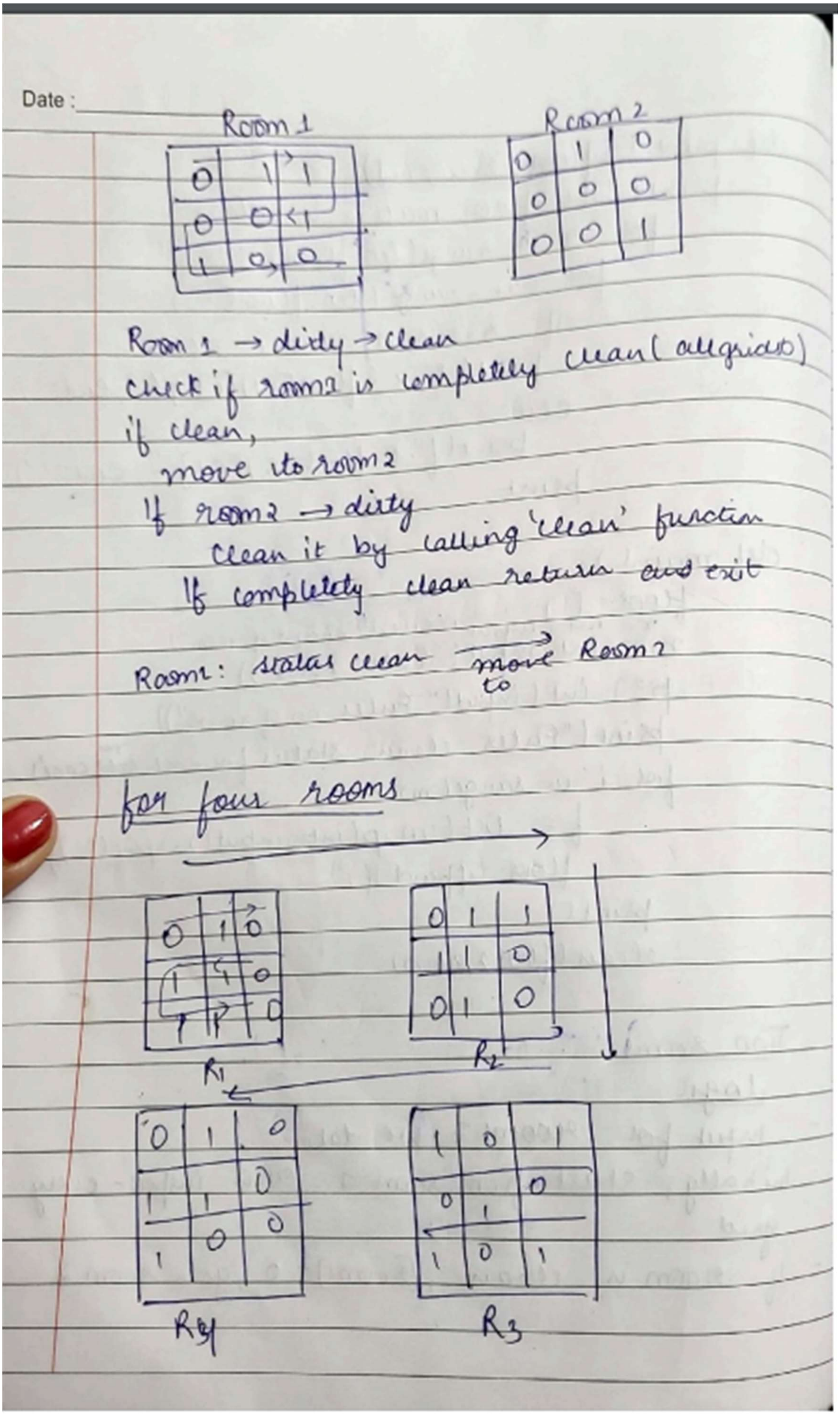
Date :

```
def print_f(floor, row, col) :
    print("The floor matrix is below")
        for r in range(len(floor)):
            for c in range(len(floor[r])):
                if r== row and c==col:
                  print(f">{floor[r][c]}<", end='')
                else:
                   print(f" {floor[r][c]} ", end='')
               print

def main():
        floor=[]
        r= int(input("Enter number of room"),
        m = int(input("Enter rows"))
         p=  int(input(" Enter no.of rows"))
         print("Enter clean status for each ~~col~~ row cell)
        for i in range(m):
              f= list(map(int,input().split(""))
               floor.append(f)
          print()
          clean(floor, room)
```

Two rooms

logic

- Input for rooms 2 we take
- Initially, start from room 1 and inspec- every grid
- If room is clean, [room1]- 0, goto room 2.



Date :

Room 1

0	1	1
0	0	1
1	0	0

Room 2

0	1	0
0	0	0
0	0	1

Room 1 → dirty → clean
check if room2 is completely clean(allgrids)
if clean,
move to room2
If room2 → dirty
Clean it by calling 'clean' function
If completely clean return and exit

Room1: status clean $\xrightarrow[\text{to}]{\text{move}}$ Room2

for four rooms

0	1	0
1	1	0
1	1	0

R1

0	1	1
1	1	0
0	1	0

R2

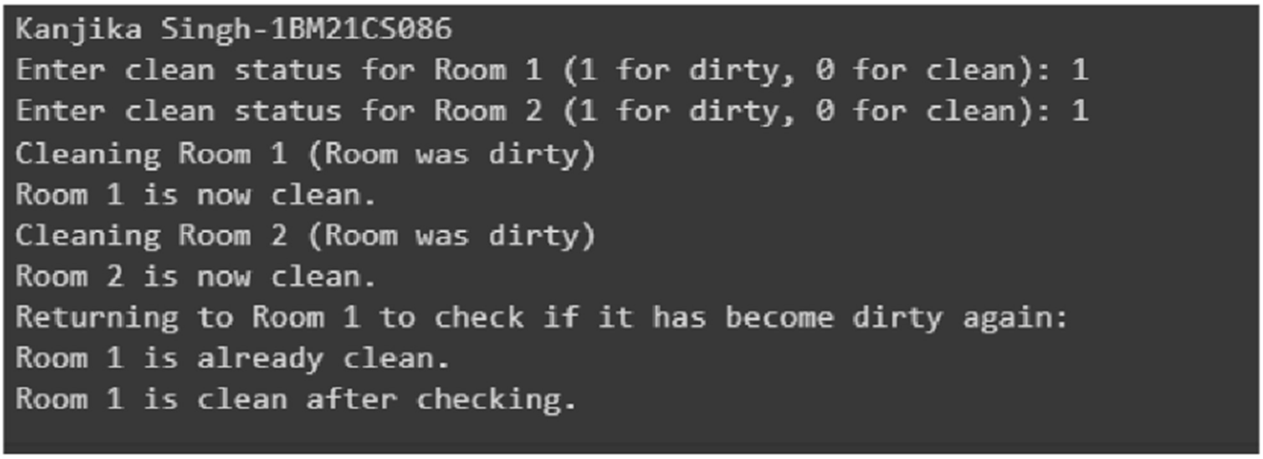
0	1	0
1	1	0
1	0	0

R4

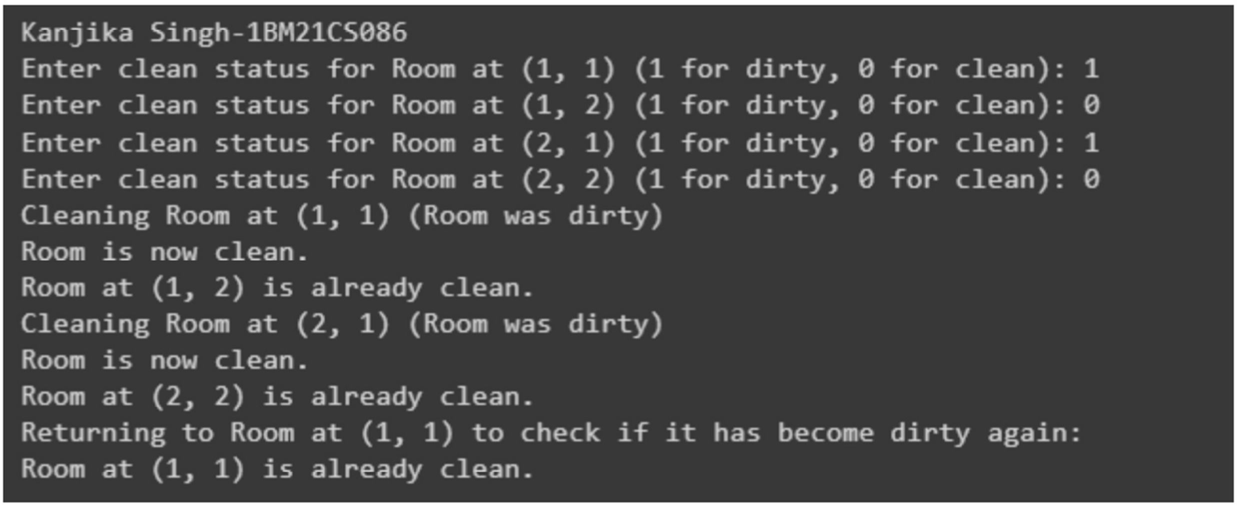
1	0	1
0	1	0
1	0	1

R3

Output:



```
Kanjika Singh-1BM21CS086
Enter clean status for Room 1 (1 for dirty, 0 for clean): 1
Enter clean status for Room 2 (1 for dirty, 0 for clean): 1
Cleaning Room 1 (Room was dirty)
Room 1 is now clean.
Cleaning Room 2 (Room was dirty)
Room 2 is now clean.
Returning to Room 1 to check if it has become dirty again:
Room 1 is already clean.
Room 1 is clean after checking.
```



```
Kanjika Singh-1BM21CS086
Enter clean status for Room at (1, 1) (1 for dirty, 0 for clean): 1
Enter clean status for Room at (1, 2) (1 for dirty, 0 for clean): 0
Enter clean status for Room at (2, 1) (1 for dirty, 0 for clean): 1
Enter clean status for Room at (2, 2) (1 for dirty, 0 for clean): 0
Cleaning Room at (1, 1) (Room was dirty)
Room is now clean.
Room at (1, 2) is already clean.
Cleaning Room at (2, 1) (Room was dirty)
Room is now clean.
Room at (2, 2) is already clean.
Returning to Room at (1, 1) to check if it has become dirty again:
Room at (1, 1) is already clean.
```