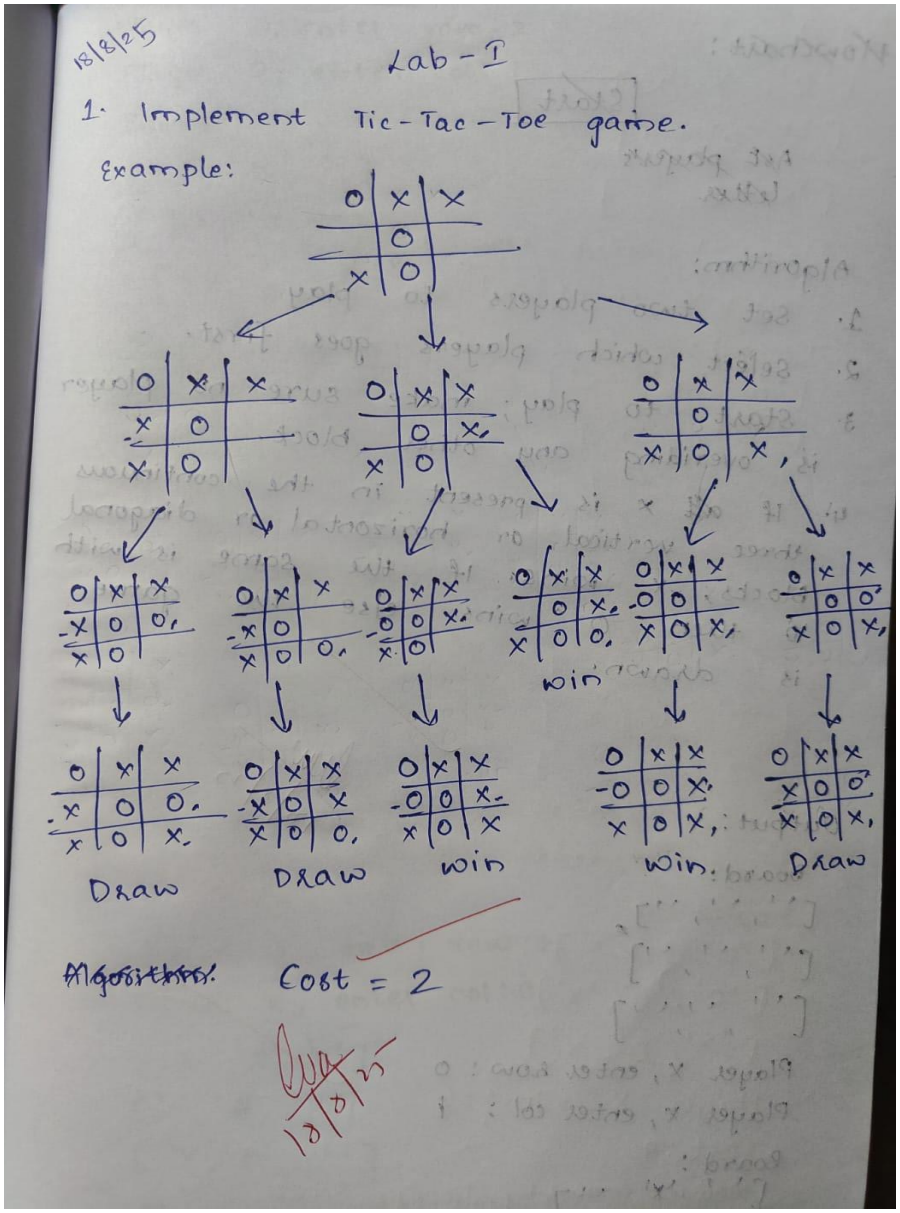
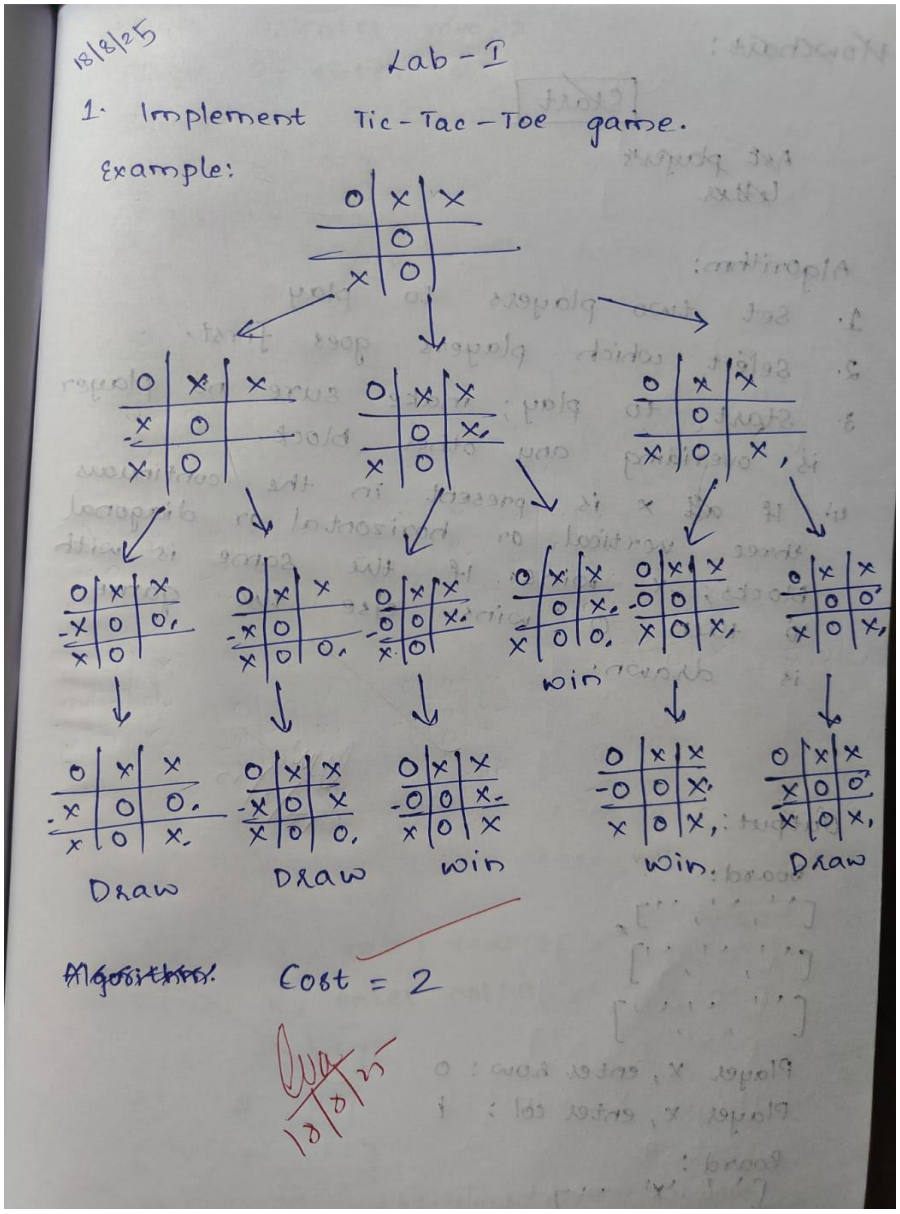
Week – 1

Program 1 Implement Tic –Tac –Toe Game

Algorithm:




18/8/25
Lab - 1
1. Implement Tic-Tac-Toe game.
Example:
win
Draw
Draw
win
win.
Draw
Algorithm:
Cost = 2
18/8/25

Output:

```
Player X, enter row (0-2): 2
Player X, enter col (0-2): 0

Current Board:
['O', 'X', 'X']
[' ', ' ', ' ']
['X', 'O', ' ']

Player O, enter row (0-2): 1
Player O, enter col (0-2): 1

Current Board:
['O', 'X', 'X']
[' ', 'O', ' ']
['X', 'O', ' ']

Player X, enter row (0-2): 1
Player X, enter col (0-2): 2

Current Board:
['O', 'X', 'X']
[' ', 'O', 'X']
['X', 'O', ' ']

Player O, enter row (0-2): 1
Player O, enter col (0-2): 0

Current Board:
['O', 'X', 'X']
['O', 'O', 'X']
['X', 'O', ' ']

Player X, enter row (0-2): 2
Player X, enter col (0-2): 2

Current Board:
['O', 'X', 'X']
['O', 'O', 'X']
['X', 'O', 'X']

Player X wins!
Total moves (cost): 9
```

```
[' ', ' ', ' ']

Player O, enter row (0-2): 2
Player O, enter col (0-2): 1

Current Board:
['O', 'X', 'X']
[' ', ' ', ' ']
[' ', 'O', ' ']

Player X, enter row (0-2): 2
Player X, enter col (0-2): 0

Current Board:
['O', 'X', 'X']
[' ', ' ', ' ']
['X', 'O', ' ']

Player O, enter row (0-2): 1
Player O, enter col (0-2): 1

Current Board:
['O', 'X', 'X']
[' ', 'O', ' ']
['X', 'O', ' ']

Player X, enter row (0-2): 1
Player X, enter col (0-2): 0

Current Board:
['O', 'X', 'X']
['X', 'O', ' ']
['X', 'O', ' ']

Player O, enter row (0-2): 2
Player O, enter col (0-2): 2

Current Board:
['O', 'X', 'X']
['X', 'O', ' ']
['X', 'O', 'O']

Player O wins!
Total moves (cost): 8
```

```
Player O, enter row (0-2): 2
Player O, enter col (0-2): 1

Current Board:
['O', 'X', 'X']
[' ', ' ', ' ']
[' ', 'O', ' ']

Player X, enter row (0-2): 2
Player X, enter col (0-2): 0

Current Board:
['O', 'X', 'X']
[' ', ' ', ' ']
['X', 'O', ' ']

Player O, enter row (0-2): 1
Player O, enter col (0-2): 1

Current Board:
['O', 'X', 'X']
[' ', 'O', ' ']
['X', 'O', ' ']

Player X, enter row (0-2): 1
Player X, enter col (0-2): 0

Current Board:
['O', 'X', 'X']
['X', 'O', ' ']
['X', 'O', ' ']

Player O, enter row (0-2): 2
Player O, enter col (0-2): 2

Current Board:
['O', 'X', 'X']
['X', 'O', ' ']
['X', 'O', 'O']

Player O wins!
Total moves (cost): 8
```

Code:

```
def print_board(board):
    print("\nCurrent Board:")
    for row in board:
        print(row)
    print()



def check_winner(board, player):
    # Check rows
    for row in board:
        if all(cell == player for cell in row):
            return True
```

```
    # Check columns
    for col in range(3):
        if all(board[row][col] == player for row in range(3)):
            return True

    # Check diagonals
    if all(board[i][i] == player for i in range(3)) or all(board[i][2 - i] == player for i in range(3)):
        return True

    return False



def tic_tac_toe():
    board = [[' ' for _ in range(3)] for _ in range(3)]
    players = ['X', 'O']
    move_count = 0

    while True:
        current_player = players[move_count % 2]
        print(f"Player {current_player}, enter row (0-2): ", end="")
        row = int(input())
        print(f"Player {current_player}, enter col (0-2): ", end="")
        col = int(input())

        # If cell is empty
        if board[row][col] == ' ':
            board[row][col] = current_player
            move_count += 1
            print_board(board)
```

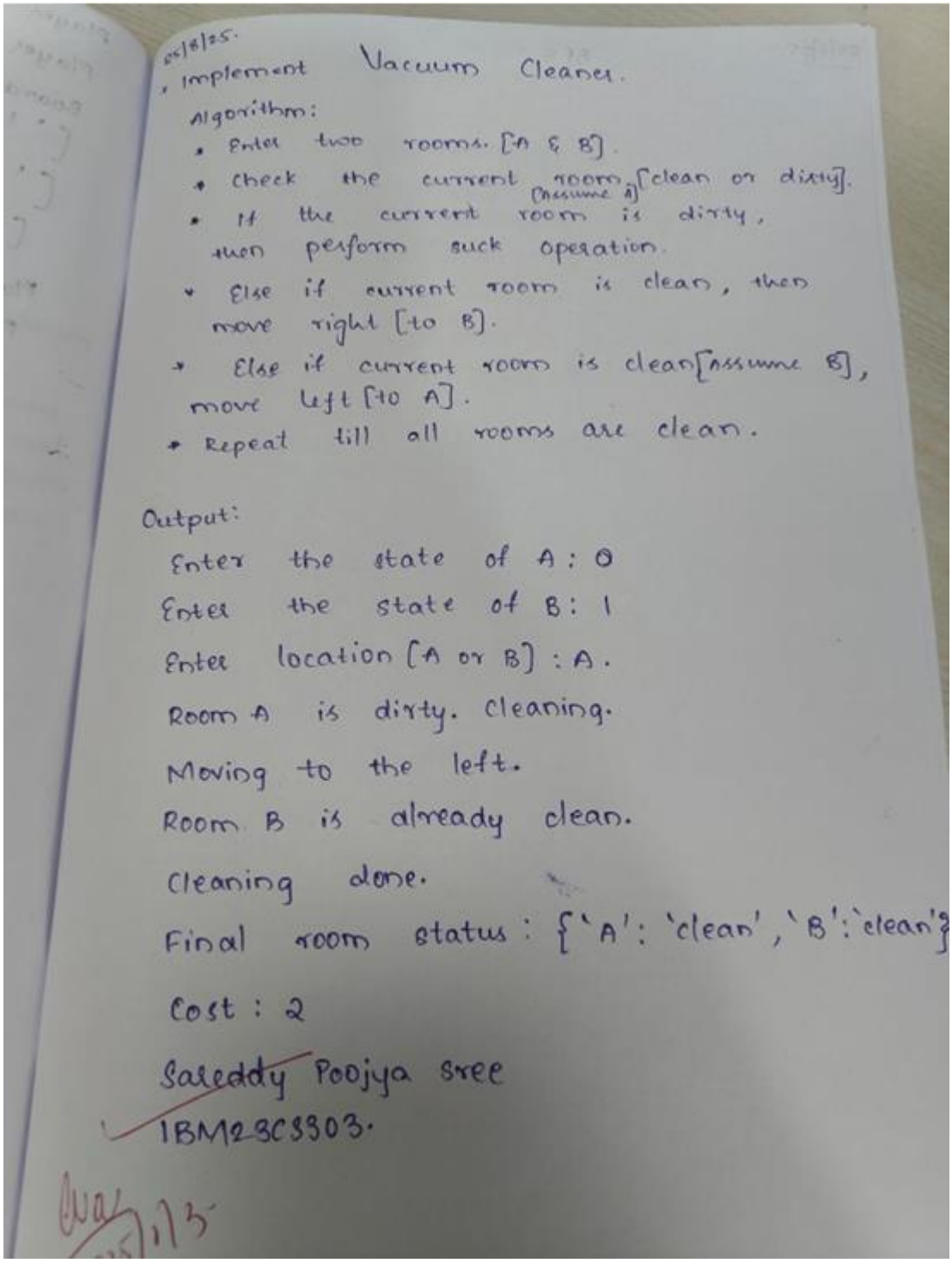
```
            if check_winner(board, current_player):
                print(f"Player {current_player} wins!")
                print(f"Total moves (cost): {move_count}")
                break

            if move_count == 9:  # Board full
                print("It's a draw!")
                break
        else:
            print("Cell already taken! Try again.")


# Run the game
tic_tac_toe()
```

b. Implement vacuum cleaner:

Algorithm:


05/8/25.
Implement Vacuum Cleaner.
Algorithm:
Enter two rooms. [A & B].
Check the current room [Assume A] [clean or dirty].
If the current room is dirty, then perform suck operation.
Else if current room is clean, then move right [to B].
Else if current room is clean [assume B], move left [to A].
Repeat till all rooms are clean.
Output:
Enter the state of A : 0
Enter the state of B : 1
Enter location [A or B] : A.
Room A is dirty. Cleaning.
Moving to the left.
Room B is already clean.
Cleaning done.
Final room status : {'A': 'clean', 'B': 'clean'}
Cost : 2
Saleddy Poojya Sree
1BM23CS303.

Output:

```
Enter status for Room A (0 = clean, 1 = dirty): 1
Enter status for Room B (0 = clean, 1 = dirty): 1
Enter starting room (A or B): A

Initial Room Statuses:
Room A: Dirty
Room B: Dirty

Vacuum starting in Room A...

Room A is dirty. Performing SUCK action.
Moving to Room B.
Room B is dirty. Performing SUCK action.

Final Room Statuses:
Room A: Clean
Room B: Clean

cost : 3
Sareddy Poojya Sree
1BM23CS303
```

```
Enter status for Room A (0 = clean, 1 = dirty): 1
Enter status for Room B (0 = clean, 1 = dirty): 0
Enter starting room (A or B): A

Initial Room Statuses:
Room A: Dirty
Room B: Clean

Vacuum starting in Room A...

Room A is dirty. Performing SUCK action.
Moving to Room B.
Room B is already clean.

Final Room Statuses:
Room A: Clean
Room B: Clean

cost : 2
Sareddy Poojya Sree
1BM23CS303
```

Code:

```
def vacuum_cleaner():
    # Input the state of rooms A and B
    state_A = int(input("Enter state of A (0 for clean, 1 for dirty): "))
    state_B = int(input("Enter state of B (0 for clean, 1 for dirty): "))
    location = input("Enter location (A or B): ").upper()

    cost = 0
    rooms = {'A': state_A, 'B': state_B}
```

```
    # Function to clean a room if dirty
    def clean_room(room):
        nonlocal cost
        if rooms[room] == 1:
            print(f"Cleaned {room}.")
            rooms[room] = 0
            cost += 1
        else:
            print(f"{room} is clean.")

    # Start cleaning based on location
    if location == 'A':
        clean_room('A')
10
        print("Moving vacuum right")
        clean_room('B')
    elif location == 'B':
        clean_room('B')
        print("Moving vacuum left")
        clean_room('A')
    else:
        print("Invalid starting location!")

    print(f"Cost: {cost}")
    print(rooms)

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