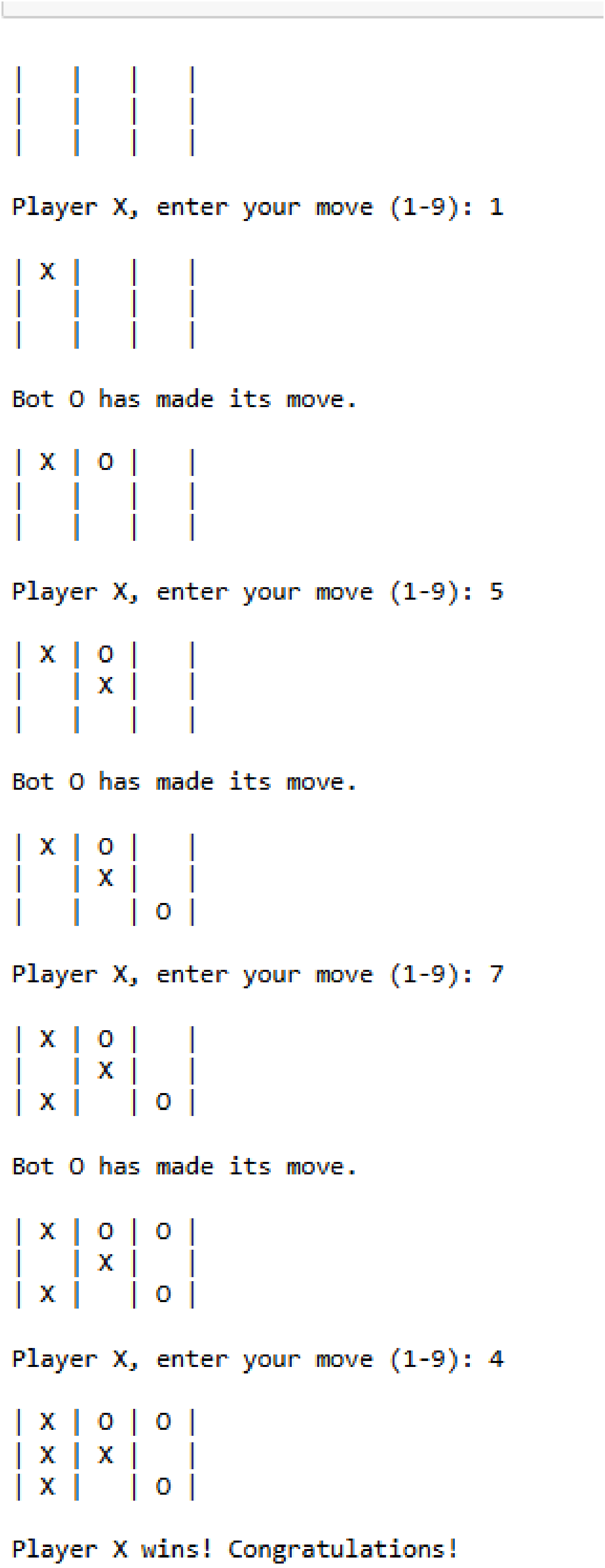
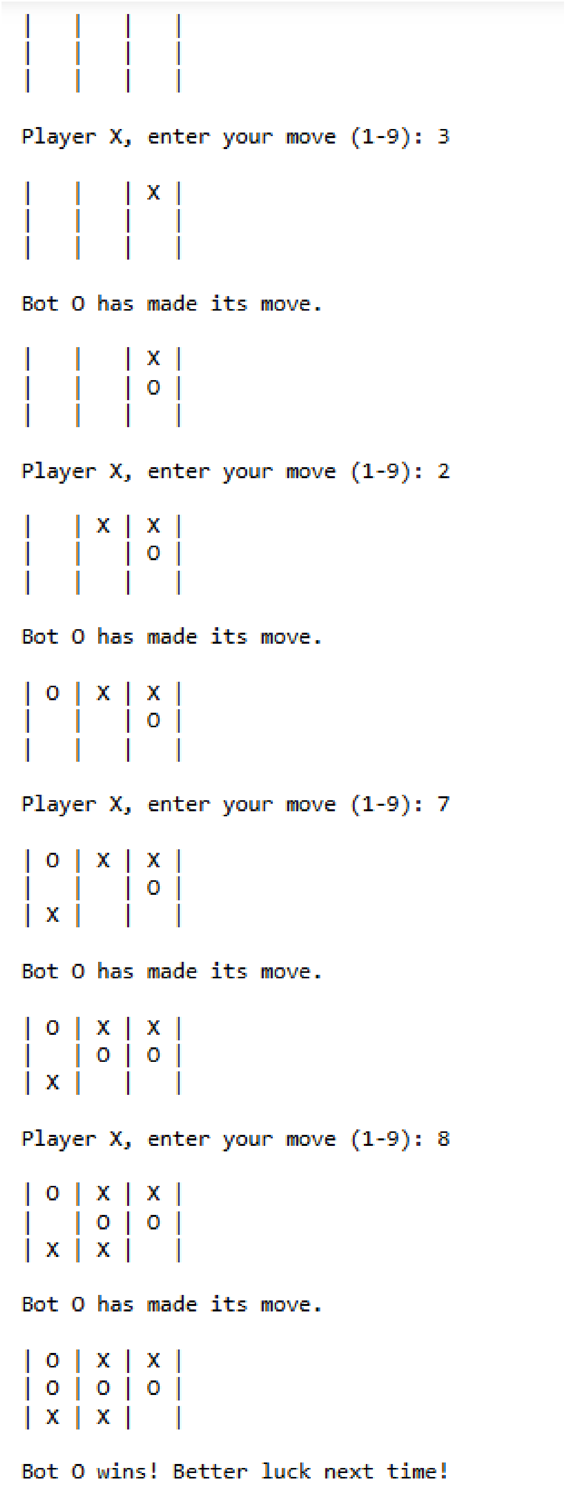
Implement Tic –Tac –Toe Game:

```
import random

board = [' ' for _ in range(9)]

def print_board():
    row1 = '| {} | {} | {} |'.format(board[0], board[1], board[2])
    row2 = '| {} | {} | {} |'.format(board[3], board[4], board[5])
    row3 = '| {} | {} | {} |'.format(board[6], board[7], board[8])

    print()
    print(row1)
    print(row2)
    print(row3)
    print()


def has_won(player):
    win_conditions = [(0, 1, 2), (3, 4, 5), (6, 7, 8), (0, 3, 6), (1, 4, 7), (2, 5, 8), (0, 4, 8), (2, 4, 6)]

    for condition in win_conditions:
        if board[condition[0]] == board[condition[1]] == board[condition[2]] == player:
            return True
    return False


def bot_move():
    for i in range(9):
        if board[i] == ' ':
            board[i] = 'O'
            if has_won('O'):
                return
            board[i] = ' '

    for i in range(9):

        if board[i] == ' ':
            board[i] = 'X'

            if has_won('X'):
                board[i] = 'O'
                return
            board[i] = ' '

    possible_moves = [i for i, x in enumerate(board) if x == ' ']

    if possible_moves:
        move = random.choice(possible_moves)
        board[move] = 'O'
    else:
        print("It's a draw!")
        exit()


def main():
    current_player = 'X'
    while True:
        print_board()
        if current_player == 'X':
            move = input("Player X, enter your move (1-9): ")

            if board[int(move) - 1] != ' ':
                print("Invalid move, try again.")
                continue
            board[int(move) - 1] = current_player

        else:
            bot_move()
            print("Bot O has made its move.")

        if has_won(current_player):
            print_board()

            if current_player == 'X':
                print("Player X wins! Congratulations!")

            else:
                print("Bot O wins! Better luck next time!")
            break

        if ' ' not in board:
            print_board()
            print("It's a draw!")
            break

        current_player = 'O' if current_player == 'X' else 'X'


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



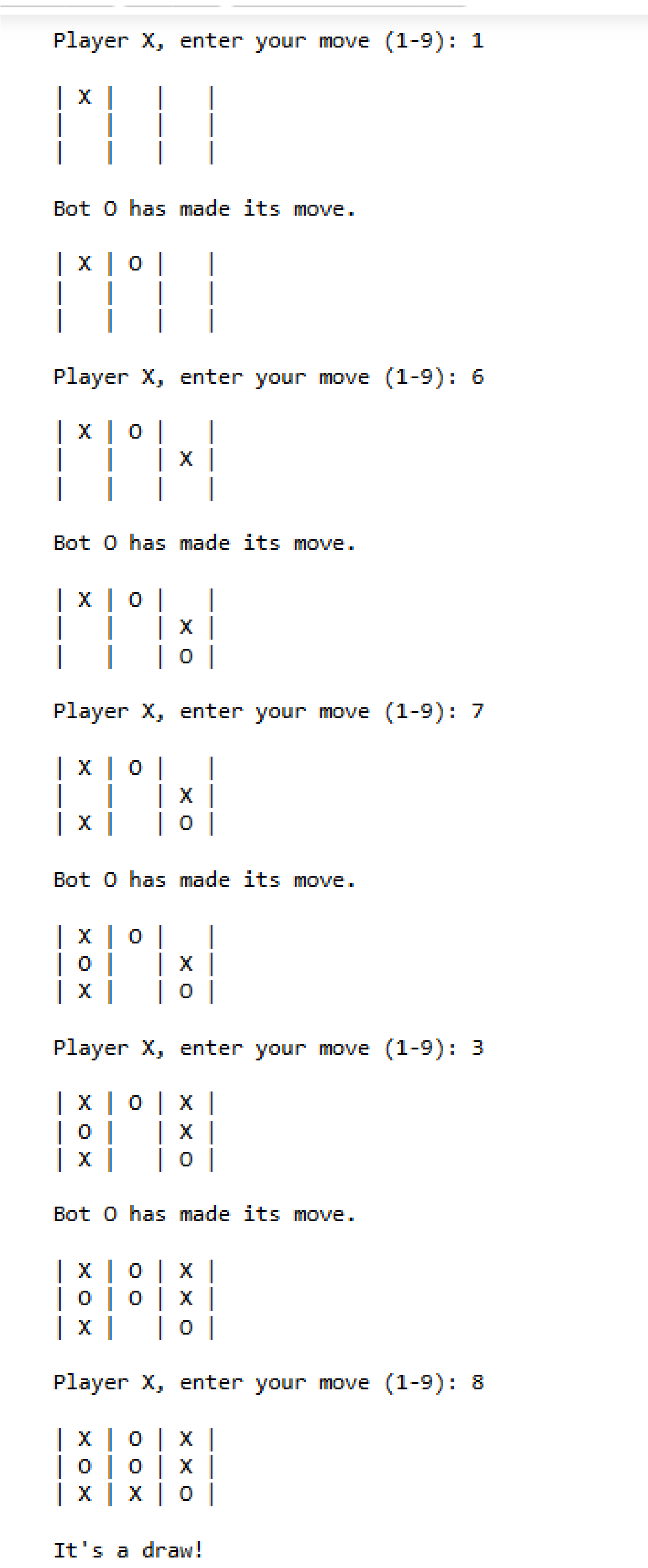
```
|   |   |   |
|   |   |   |
|   |   |   |

Player X, enter your move (1-9): 1

| X |   |   |
|   |   |   |
|   |   |   |

Bot O has made its move.

| X | O |   |
|   |   |   |
|   |   |   |

Player X, enter your move (1-9): 5

| X | O |   |
|   | X |   |
|   |   |   |

Bot O has made its move.

| X | O |   |
|   | X |   |
|   |   | O |

Player X, enter your move (1-9): 7

| X | O |   |
|   | X |   |
| X |   | O |

Bot O has made its move.

| X | O | O |
|   | X |   |
| X |   | O |

Player X, enter your move (1-9): 4

| X | O | O |
| X | X |   |
| X |   | O |

Player X wins! Congratulations!
```

Fig-1.1 OUTPUT



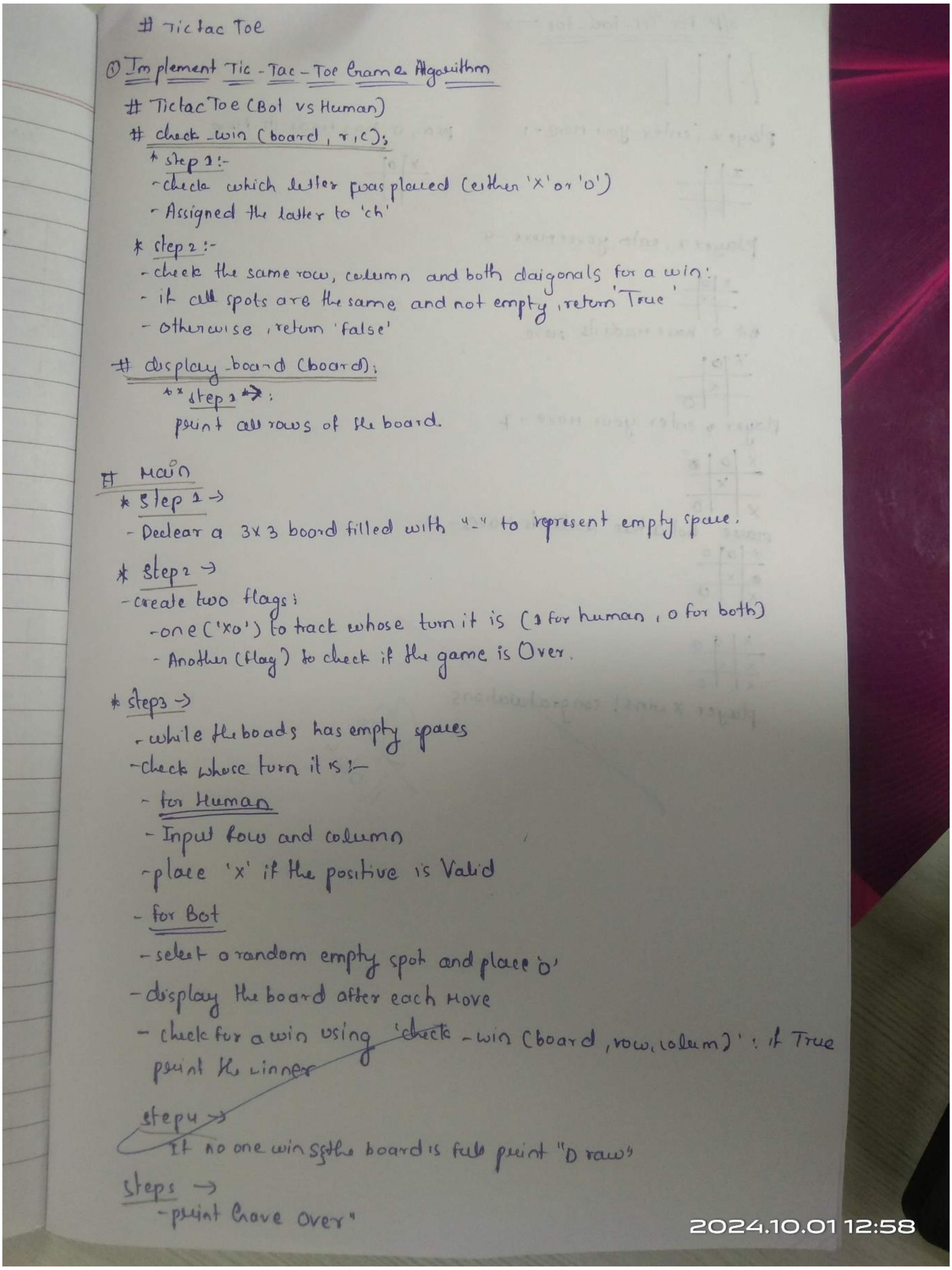
```
|   |   |   |
|   |   |   |
|   |   |   |

Player X, enter your move (1-9): 3

|   |   | X |
|   |   |   |
|   |   |   |

Bot O has made its move.

|   |   | X |
|   |   | O |
|   |   |   |

Player X, enter your move (1-9): 2

|   | X | X |
|   |   | O |
|   |   |   |

Bot O has made its move.

| O | X | X |
|   |   | O |
|   |   |   |

Player X, enter your move (1-9): 7

| O | X | X |
|   |   | O |
| X |   |   |

Bot O has made its move.

| O | X | X |
|   | O | O |
| X |   |   |

Player X, enter your move (1-9): 8

| O | X | X |
|   | O | O |
| X | X |   |

Bot O has made its move.

| O | X | X |
| O | O | O |
| X | X |   |

Bot O wins! Better luck next time!
```

Fig-1.2 OUTPUT



```
Player X, enter your move (1-9): 1

| X |   |   |
|   |   |   |
|   |   |   |

Bot O has made its move.

| X | O |   |
|   |   |   |
|   |   |   |

Player X, enter your move (1-9): 6

| X | O |   |
|   |   | X |
|   |   |   |

Bot O has made its move.

| X | O |   |
|   |   | X |
|   |   | O |

Player X, enter your move (1-9): 7

| X | O |   |
|   |   | X |
| X |   | O |

Bot O has made its move.

| X | O |   |
| O |   | X |
| X |   | O |

Player X, enter your move (1-9): 3

| X | O | X |
| O |   | X |
| X |   | O |

Bot O has made its move.

| X | O | X |
| O | O | X |
| X |   | O |

Player X, enter your move (1-9): 8

| X | O | X |
| O | O | X |
| X | X | O |

It's a draw!
```

Fig-1.3 OUTPUT



Tictac Toe

① Implement Tic-Tac-Toe Grame Algorithm

TictacToe (Bot vs Human)

check_win (board, r, c);

* step 1:-
  - check which letter was placed (either 'X' or 'O')
  - Assigned the latter to 'ch'

* step 2:-
  - check the same row, column and both daigonals for a win.
  - if all spots are the same and not empty, return 'True'
  - otherwise, return 'false'

display_board (board);

** step 1 ** :
print all rows of the board.

Main

* Step 1 →
  - Declear a 3x3 boord filled with "-" to represent empty space.

* step 2 →
  - create two flags:
    - one ('xo') to track whose turn it is (1 for human, 0 for both)
    - Another (flag) to check if the game is Over.

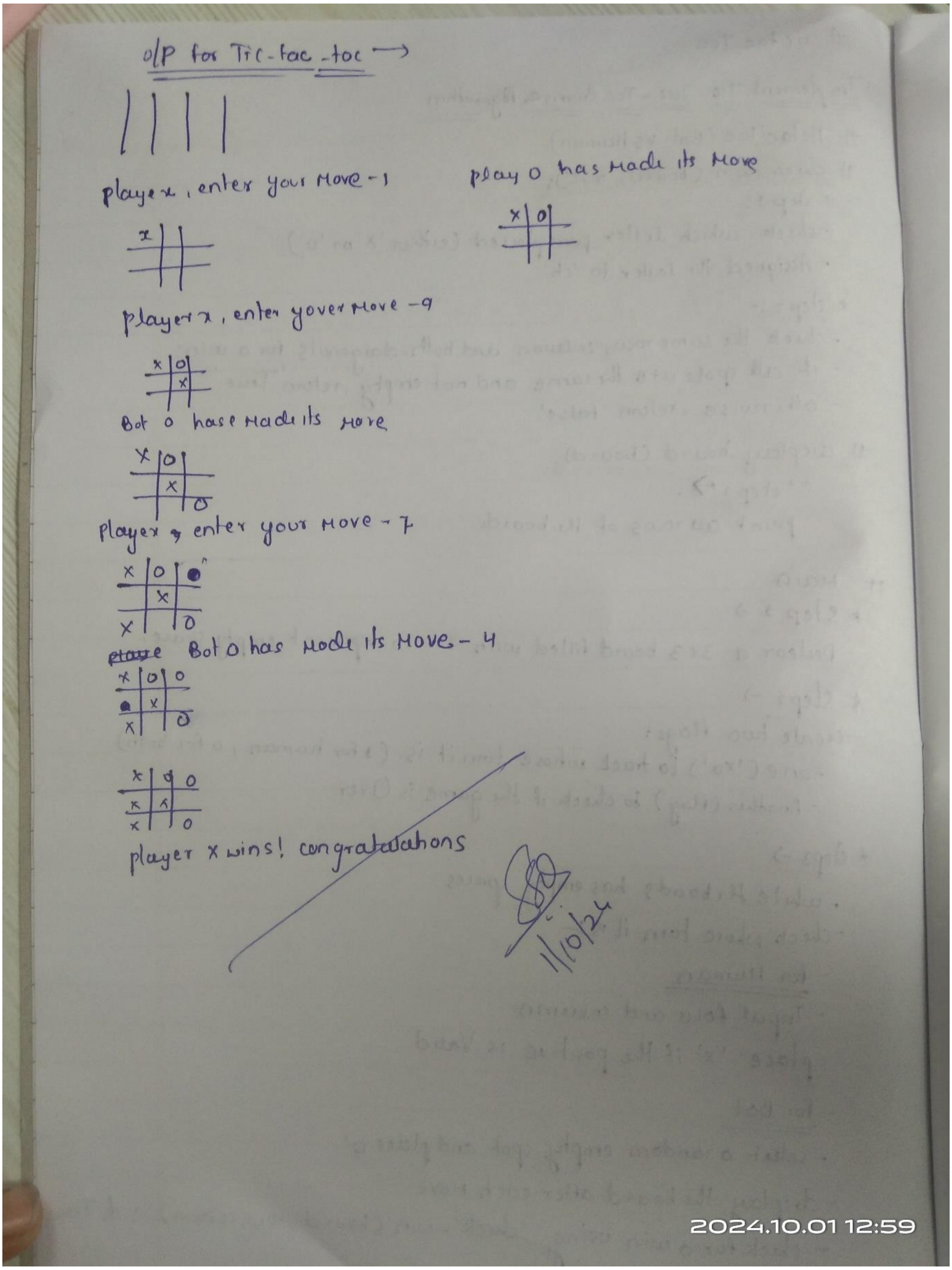
* step 3 →
  - while the boads has empty spaces
  - Check whose turn it is:-
    - for Human
    - Input row and column
    - place 'x' if the positive is Valid
  - for Bot
    - select a random empty spot and place 'O'
  - display the board after each Move
  - check for a win using 'check_win (board, row, colum)': if True print the winner

step 4 →
  If no one wins & the board is full print "D raw"

step 5 →
  - print Gave Over"

2024.10.01 12:58

Fig-2.1 Observation



o/p for Tic-tac-toe →

```
 | | |

player x, enter your move - 1

 x | |
---+-+---
   | |
---+-+---
   | |

play 0 has made its move

 x | 0 |
---+---+---
   |   |
---+---+---
   |   |

player x, enter yover move - 9

 x | 0 |
---+---+---
   | x |
---+---+---
   |   |

Bot 0 hase made its move

 x | 0 |
---+---+---
   | x |
---+---+---
   |   | 0

Player x, enter your move - 7

 x | 0 | ●
---+---+---
   | x |
---+---+---
 x |   | 0
```

~~plaxe~~ Bot 0 has mode its move - 4

```
 x | 0 | 0
---+---+---
 ● | x |
---+---+---
 x |   | 0

 x | 0 | 0
---+---+---
 x | x |
---+---+---
 x |   | 0

player x wins! congratulations
```

1/10/24

2024.10.01 12:59

Fig-2.2 Observation