# LAB-1 . Tik-Tac-Toe

## Algorithm

```
1. Initialize
   -> create a 3x3 grid filled with empty spaces
      (' ').
   -> Let human player choose between 'x' or 'o'
   board ← [[' ', ' ', ' '], [' ', ' ', ' '], [' ', ' ', ' ']]
   player ← 'x' or 'o'
   computer ← 'o' if player == 'x' else 'x'

2. Repeat until win or draw:
3. Display the board.
4. Human Player Move.
   -> Let user select a position (1-9)
   -> validate the move
   move ← input("Enter move (1-9): ")
   row, col ← divmod(move-1, 3)
    if move not in range(1, 10) or board[row][col] != ' ':
          print("Invalid move. Try again")
          continue
    else:
          board[row][col] ← player
5. Check for win after the user move
    if check_win(board, player):
        print_board(board)
        print("Player {player} wins!")
        break.
6. Check for draw.
```

```
7. Computer Move.
      empty_cells ← [(row, col) for row in range(3) for
      col in range(3) if board[row][col] == ' ']
      if empty_cells:
         row, col ← random.choice(empty_cells)
         board[row][col] ← Computer,
8. check for win
9. check for draw
10. Repeat.
```

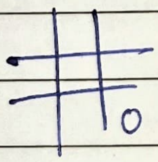
<u>Code:-</u>

```
import random
def initialize_board():
    return [[' ' for _ in range(3)] for _ in range(3)]
def display_board(board):
    for row in board:
        print('|'.join(row))
        print('-' * 5)
def check_winner(board):
    for row in board:
        if row[0] == row[1] == row[2] != ' ':
        return row[0]
    for col in range(3):
        if board[0][col] == board[1][col] == board[2][col] = ' ':
            return board[0][col]
    if board[0[0] == board[1][1] == board[2][2] = '
        return board[0][0]
    if board[0][2] == board[1][1] == board[2][0] != '
        return board[0][2]
        return None
def available_moves(board):
    return[(i, j) for i in range(3) for j in range(3) if
      board[i][j]==' ']
def check_two_in_a_row(board, player):
    for row in range(3):
        if board[row].count(player)==2 and board[row].
        count(' ')==1:
            return row, board[row].index(' ')
    for col in range(3):
        if[board[row][col] for row in range(3)].
        count(player)==2:
            empty_index=[row for row in range(3)
                    if board[row][col] ==' ']
            if empty_index:
                return empty_index[0], col
    if[board[i][i] for i in range(3)) count(player)==2
        empty_index = [i for i in range(3) if board[i][i]
            ==' ']
        if empty_index:
            return empty_index[0], empty_index[0]
=' ':
    if [board[i][2-i] for i in range(3)]. Count(player)==2:
        empty_index =[i for i in range(3) if board[i][2-i]
                                              ==' ']
        if empty_index:
            return empty_index[0], 2-empty_index[0]
    return None.
def make_move(board, player, move):
    board[move[0]][move [1]] = player
```

```
def computer_move(board):
    move=check_two_in_a_row(board,'O')
    if move:
        make_move(board,'O',move)
        return.
    move=Check_two_in_a_row(board,'x')
    if move:
        make_move(board,'O', move)
        return.
    moves =available_moves(board)
    if moves:
        move=random.Choice(moves)
        make_move(board,'O', move)
def user_move(board):
    while True:
        try:
            row =int(int(input("Enter row(0-2): "))
            col = int(input("Enter column (0-2): "))
            if board[row][col]==' ':
                make_move(board,'x', (row,col))
                return
            else:
                print("That spot is already taken. Try again.")
        except(ValueError, IndexError):
            print("Invalid input. Please enter numbers between
                0 & 2.")
def play_game():
    board = initialize_board()
    Players=['x','o']
    current_player=0
    for_in range(9):
        display_board(board)
        if current_player ==0:
            user_move(board)
        else:
            computer_move(board)
        winner= check_winner(board)
        if winner:
            display_board(board)
            print("Player {winner} wins!")
            return
        current_player=1-current_player
    display_board(board)
    print("It's a draw!")
play_game()
```

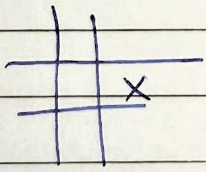
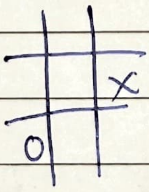
[illegible]
25/9/24

Output:



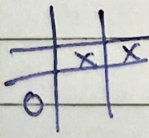
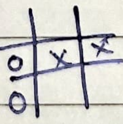
Enter row: 1
col: 2

again,")

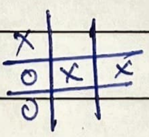
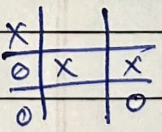



between

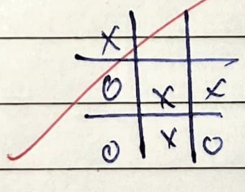
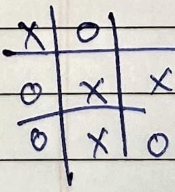
Enter row: 1
col: 1

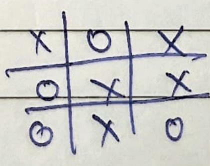
Enter row: 0
col: 0

Enter row: 2
col: 1

row: 0, col: 2



it's a draw