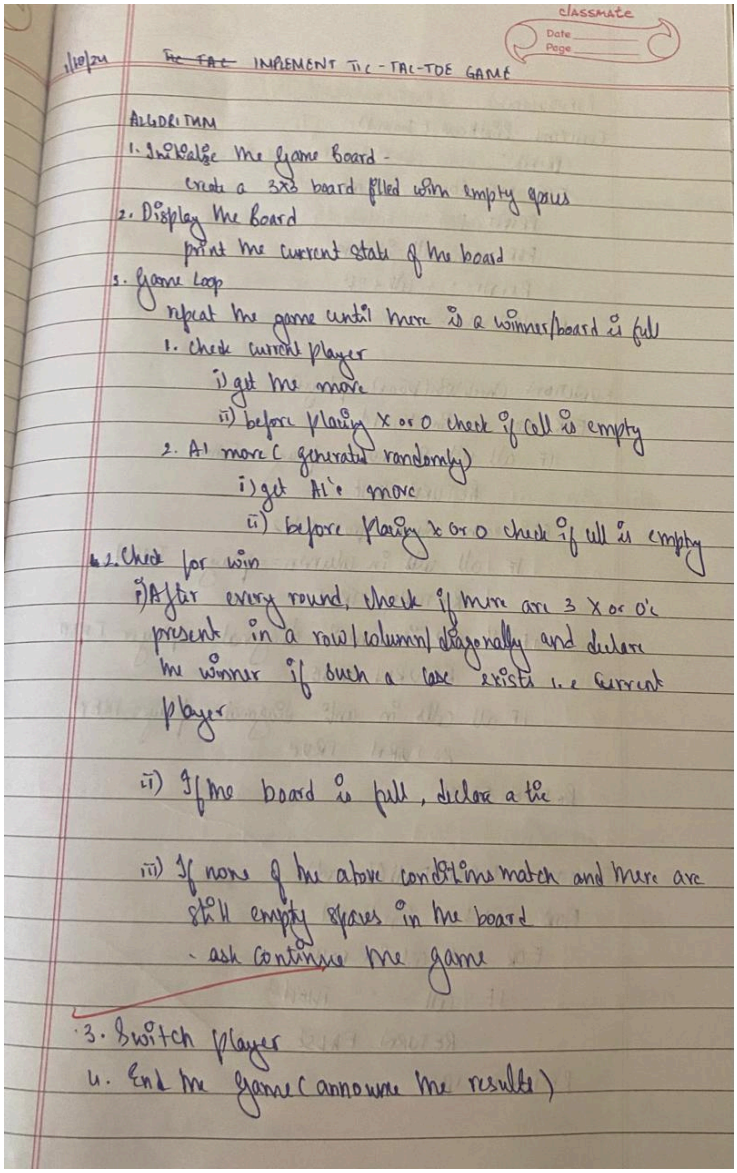
Date:1/10/24

Program Title:Implement Tic Tac Toe Game

Algorithm -



1/10/24 ~~To PAT~~ IMPLEMENT TIC-TAC-TOE GAME

ALGORITHM

1. Initialize the Game Board -
   create a 3x3 board filled with empty spaces
2. Display the Board
   print the current state of the board
3. Game Loop
   repeat the game until there is a winner/board is full
   1. check current player
      i) get the move
      ii) before placing X or O check if cell is empty
   2. AI move (generated randomly)
      i) get AI's move
      ii) before placing X or O check if ell is empty
4. Check for win
   i) After every round, check if there are 3 X or O's present in a row/column/diagonally and declare the winner if such a case exists i.e current player

   ii) If the board is full, declare a tie

   iii) If none of the above conditions match and there are still empty spaces in the board
   - ask continue the game

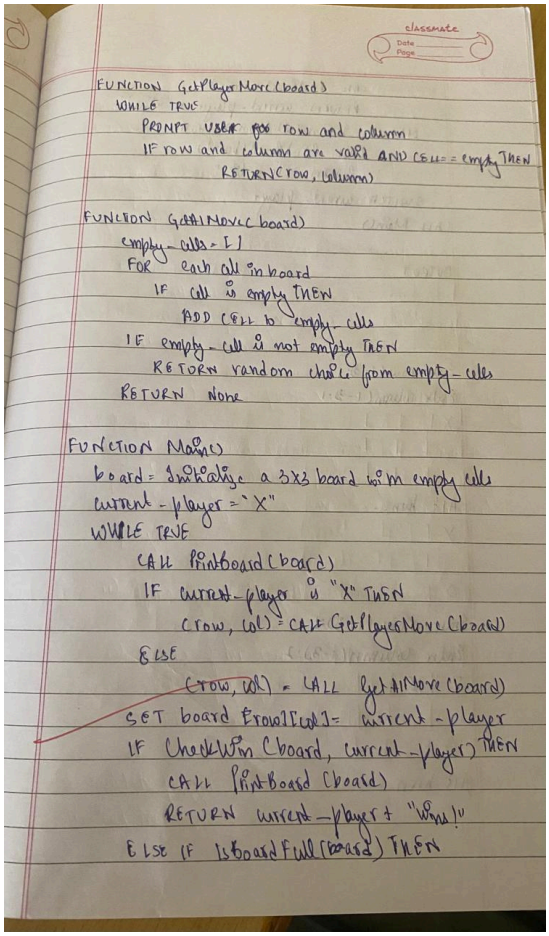
3. Switch player
4. End the game (announce the results)

PSEUDOCODE

```
FUNCTION PrintBoard(board)
    PRINT "- - - - -"
    FOR each row in board
        PRINT "|"
        FOR each cell in row
        PRINT " " + cell + "|"
    PRINT "|- - - - -"

FUNCTION CheckWin(board, player)
    FOR each row in board
        IF all cells in row == player THEN
            RETURN TRUE
    FOR each column from 0 to 2
        IF all cells in column == player THEN
            RETURN TRUE
    IF all cells in main diagonal == player THEN
            RETURN TRUE
    IF all cells in anti-diagonal == player THEN
            RETURN TRUE
    RETURN FALSE

FUNCTION IsBoardFull(board)
    FOR each row in board
        FOR each cell in row
            IF cell == " " THEN
                RETURN FALSE
    RETURN TRUE
```



FUNCTION GetPlayerMove(board)
  WHILE TRUE
    PROMPT USER for row and column
    IF row and column are valid AND CELL == empty THEN
      RETURN(row, column)

FUNCTION GetAIMove(board)
  empty_cells = []
  FOR each cell in board
    IF cell is empty THEN
      ADD CELL to empty_cells
  IF empty_cell is not empty THEN
    RETURN random choice from empty_cells
  RETURN None

FUNCTION Main()
  board = Initialize a 3x3 board with empty cells
  current_player = "X"
  WHILE TRUE
    CALL PrintBoard(board)
    IF current_player is "X" THEN
      (row, col) = CALL GetPlayerMove(board)
    ELSE
      (row, col) = CALL GetAIMove(board)
    SET board[row][col] = current_player
    IF CheckWin(board, current_player) THEN
      CALL PrintBoard(board)
      RETURN current_player + "Wins!"
    ELSE IF IsBoardFull(board) THEN

CALL PrintBoard(board)
RETURN current_player + "wins!"
ELSE IF IsBoardFull(board) THEN
CALL PrintBoard(board)
RETURN "It's a tie!"
SWITCH current_player
CALL Main()

OUTPUT

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

Enter row (1-3): 1
Enter column (1-3): 1

```
|X| | |
| | | |
| | | |
```

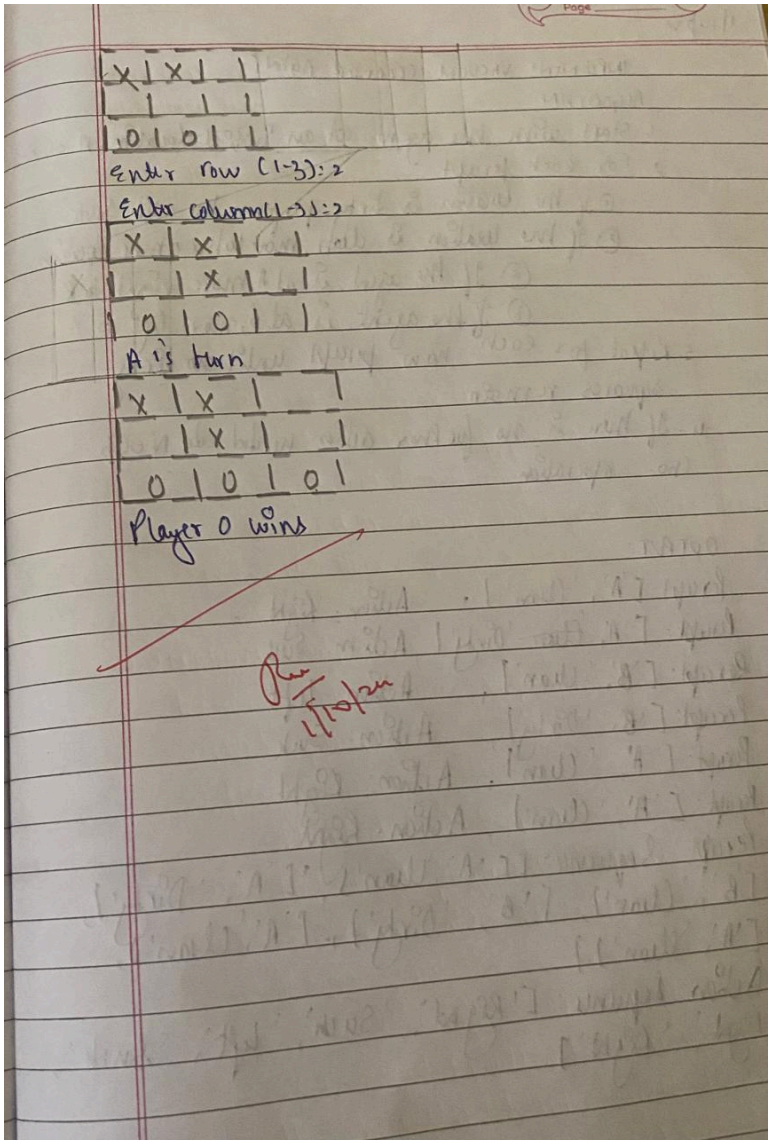
AI's turn...

```
|X| | |
| | | |
|O| | |
```

Enter row (1-3): 1
Enter column (1-3): 2

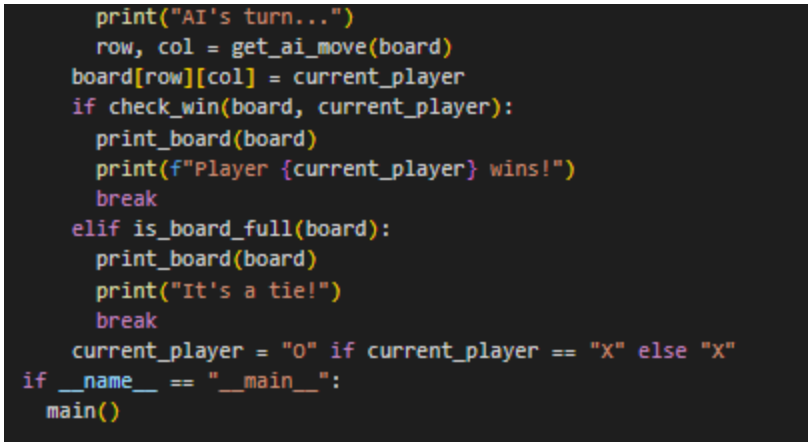
```
|X|X| |
| | | |
|O| | |
```

AI's turn...


X | X |
| |
0 | 0 |
Enter row (1-3): 2
Enter column(1-3): 2
X | X |
| X |
0 | 0 |
A's turn
X | X |
| X |
0 | 0 | 0
Player 0 wins

Code:

```
[5]  import random
     def print_board(board):
       print("-------------")
       for row in board:
         print("|", end="")
         for cell in row:
           print(" " + cell + " |", end="")
         print("\n-------------")
     def check_win(board, player):
       for row in board:
         if all(cell == player for cell in row):
           return True
       for col in range(3):
         if all(board[row][col] == player for row in range(3)):
           return True
       if all(board[i][i] == player for i in range(3)):
         return True
       if all(board[i][2 - i] == player for i in range(3)):
         return True
       return False
     def is_board_full(board):
       for row in board:
         for cell in row:
           if cell == " ":
             return False
       return True
     def get_player_move(board):
       while True:
         try:
           row = int(input("Enter row (1-3): ")) - 1
           col = int(input("Enter column (1-3): ")) - 1
           if 0 <= row <= 2 and 0 <= col <= 2 and board[row][col] == " ":
             return row, col
           else:
             print("Invalid move. Try again.")
         except ValueError:
           print("Invalid input. Enter numbers only.")
     def get_ai_move(board):
       empty_cells = []
       for row in range(3):
         for col in range(3):
           if board[row][col] == " ":
             empty_cells.append((row, col))
       if empty_cells:
         return random.choice(empty_cells)
       return None
     def main():
       board = [[" " for _ in range(3)] for _ in range(3)]
       current_player = "X"
       while True:
         print_board(board)
         if current_player == "X":
           row, col = get_player_move(board)
         else:
           print("AI's turn...")
```



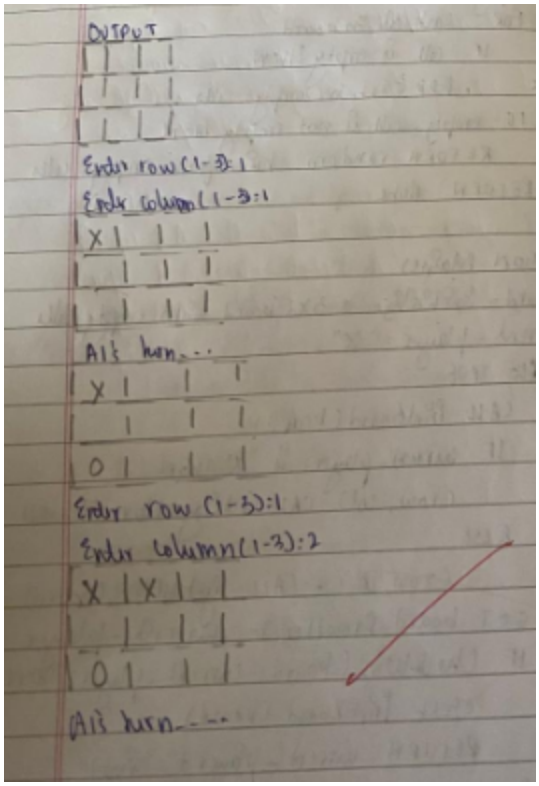
```
      print("AI's turn...")
      row, col = get_ai_move(board)
    board[row][col] = current_player
    if check_win(board, current_player):
      print_board(board)
      print(f"Player {current_player} wins!")
      break
    elif is_board_full(board):
      print_board(board)
      print("It's a tie!")
      break
    current_player = "O" if current_player == "X" else "X"
if __name__ == "__main__":
  main()
```

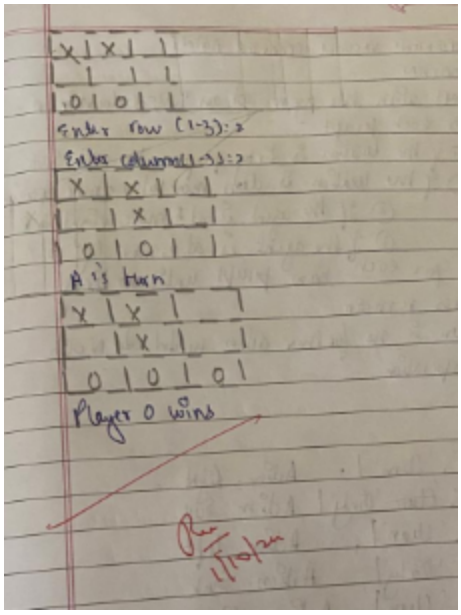
```
import random
def print_board(board):
  print("-------------")
  for row in board:
    print("|", end="")
    for cell in row:
      print(" " + cell + " |", end="")
    print("\n-------------")
def check_win(board, player):
  for row in board:
    if all(cell == player for cell in row):
      return True
  for col in range(3):
    if all(board[row][col] == player for row in range(3)):
      return True
  if all(board[i][i] == player for i in range(3)):
    return True
  if all(board[i][2 - i] == player for i in range(3)):
    return True
  return False
def is_board_full(board):
  for row in board:
    for cell in row:
      if cell == " ":
        return False
  return True
def get_player_move(board):
  while True:
    try:
      row = int(input("Enter row (1-3): ")) - 1
      col = int(input("Enter column (1-3): ")) - 1
      if 0 <= row <= 2 and 0 <= col <= 2 and board[row][col] == " ":
```

```
        return row, col
      else:
        print("Invalid move. Try again.")
    except ValueError:
      print("Invalid input. Enter numbers only.")
def get_ai_move(board):
  empty_cells = []
  for row in range(3):
    for col in range(3):
      if board[row][col] == " ":
        empty_cells.append((row, col))
  if empty_cells:
    return random.choice(empty_cells)
  return None
def main():
  board = [[" " for _ in range(3)] for _ in range(3)]
  current_player = "X"
  while True:
    print_board(board)
    if current_player == "X":
      row, col = get_player_move(board)
    else:
      print("AI's turn...")
      row, col = get_ai_move(board)
    board[row][col] = current_player
    if check_win(board, current_player):
      print_board(board)
      print(f"Player {current_player} wins!")
      break
    elif is_board_full(board):
      print_board(board)
      print("It's a tie!")
      break
    current_player = "O" if current_player == "X" else "X"
if __name__ == "__main__":
  main()
```

Snapshot of the output: (NEXT PAGE)

```
-------------
|   |   |   |
-------------
|   |   |   |
-------------
|   |   |   |
-------------
Enter row (1-3): 1
Enter column (1-3): 1
-------------
| X |   |   |
-------------
|   |   |   |
-------------
|   |   |   |
-------------
AI's turn...
-------------
| X |   |   |
-------------
|   |   |   |
-------------
| O |   |   |
-------------
Enter row (1-3): 1
Enter column (1-3): 2
-------------
| X | X |   |
-------------
|   |   |   |
-------------
| O |   |   |
-------------
AI's turn...
-------------
| X | X |   |
-------------
|   |   |   |
-------------
| O | O |   |
-------------
Enter row (1-3): 2
Enter column (1-3): 2
-------------
| X | X |   |
-------------
|   | X |   |
-------------
| O | O |   |
-------------
AI's turn...
-------------
| X | X |   |
-------------
|   | X |   |
-------------
| O | O | O |
-------------
Player O wins!
```





STATE SPACE TREE: