**Name: Usman Ul Haq**

**Roll# BSAIM-035**

**AI LAB TASKs**

**Documentation**

**Write code for Min Max Algorithm**

This is a **Tic-Tac-Toe AI using the Minimax algorithm**. Here’s a breakdown of what the code does:

**Overview**

* The game is played on a **3x3 board** represented as a **list of 9 elements**.
* The **Minimax algorithm** is used to determine the best move for the AI (**O**).
* The player (**X**) enters their moves manually.
* The game continues in a loop until **X wins, O wins, or it's a draw**.

**Function Breakdown**

**1. is\_winner(board, player)**

* Checks if a given player (**X or O**) has won.
* It compares the board's state against **all possible winning combinations**.

**2. is\_draw(board)**

* Checks if all board positions are filled (i.e., no spaces left).
* If the board is full and there's no winner, it's a draw.

**3. minimax(board, depth, is\_maximizing)**

* Implements the **Minimax algorithm**:
  + If O wins, return **+1**.
  + If X wins, return **-1**.
  + If it's a draw, return **0**.
  + If it's O's turn (maximizing player), it **chooses the move with the highest score**.
  + If it's X's turn (minimizing player), it **chooses the move with the lowest score**.
  + Recursively evaluates all possible board states.

**4. best\_move(board)**

* Finds the best possible move for the AI (O).
* Calls the minimax function to evaluate the best option.
* Returns the **index** of the best move.

**5. print\_board(board)**

* Prints the board in a **human-readable** format.

**Game Loop**

1. The board is displayed.
2. It checks if **X** or **O** has won, or if it’s a draw.
3. The **player (X) inputs a move** (0-8).
4. If the move is valid, it's placed on the board.
5. The AI (**O**) then makes its move using the best\_move() function.
6. The loop continues until there's a **winner or a draw**.