IMPLEMENTING TIC-TAC-TOE WITH A MINIMAX ALGORITHM

In this assignment, you'll implement an AI player for the classic game of Tic-Tac-Toe. You'll be using the Minimax algorithm to create an AI that always makes optimal decisions in the game. By the end of this exercise, you'll have a working AI that you can play against using the provided graphical interface (`runner.py`). In order to do this, you'll complete the `tictactoe.py` file and implement the required functions. These functions define the game logic and AI decision-making.

Let's have a look at each function:

- `initial_state()`: Create this function to return the initial empty game board, where all cells are set to `EMPTY`.
- `player(board)`: This function determines whose turn it is to play (either "X" or "O").
- `actions(board)`: This function returns a set of possible actions (cell positions) that a player can take on the given board. You need to iterate through the board and identify the empty cells (cells containing `EMPTY`). For each empty cell, add its position as a tuple (row, column) to the set of possible actions.
- `result(board, action)`: Given a board and an action, this function creates a new board after applying the action. You need to create a deep copy of the original board and update the cell.
- `winner(board)`: This function checks if there's a winner in the current board state. You need to check all rows, columns, and diagonals to see if they have three matching symbols (either "X" or "O"). If a winning pattern is found, return the corresponding symbol; otherwise, return `None`.
- `terminal(board)`: This function checks if the game is over, either due to a winner or a tie.
- `utility(board)`: This function assigns a utility value to a terminal board state. If "X" wins, return 1; if "O" wins, return -1; if it's a tie, return 0.
- `minimax(board)`: The heart of the AI! This function implements the Minimax algorithm to find the optimal move for the current player. You'll need

to consider whether it's "X" or "O" turn and recursively evaluate the possible future game states using the `max_value` and `min_value` functions.

- `max_value(board)` and `min_value(board)`: These functions evaluate the possible future game states for both players. They will iterate through possible actions and recursively call each other.

Use the `runner.py` graphical interface to test your AI against human players.

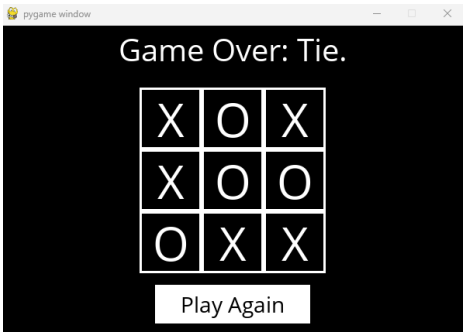

Figure 1: Pygame interface for a game of tic-tac-toe.