//tictactoe AI

import java.util.Scanner;

public class TicTacToe\_AI {

    public static void main(String[] args) {

        System.out.println("Welcome to Tic-Tac-Toe!");

        playGame();

    }

    // Function to print the Tic-Tac-Toe board with row and column indices

    public static void printBoard(char[][] board) {

        System.out.println("  0 1 2"); // Column indices

        for (int i = 0; i < 3; ++i) {

            System.out.print(i + " "); // Row index

            for (int j = 0; j < 3; ++j) {

                System.out.print(board[i][j] + " ");

            }

            System.out.println();

        }

    }

    // Function to check if a player has won

    public static boolean checkWin(char[][] board, char player) {

        // Check rows and columns

        for (int i = 0; i < 3; ++i) {

            if ((board[i][0] == player && board[i][1] == player && board[i][2] == player) ||

                (board[0][i] == player && board[1][i] == player && board[2][i] == player)) {

                return true;

            }

        }

        // Check diagonals

        if ((board[0][0] == player && board[1][1] == player && board[2][2] == player) ||

            (board[0][2] == player && board[1][1] == player && board[2][0] == player)) {

            return true;

        }

        return false;

    }

    // Function to check if the board is full

    public static boolean isBoardFull(char[][] board) {

        for (int i = 0; i < 3; ++i) {

            for (int j = 0; j < 3; ++j) {

                if (board[i][j] == ' ')

                    return false;

            }

        }

        return true;

    }

    // Function to evaluate the board for the minimax algorithm

    public static int evaluateBoard(char[][] board) {

        // Check if the AI has won

        if (checkWin(board, 'O')) {

            return 1;

        }

        // Check if the player has won

        else if (checkWin(board, 'X')) {

            return -1;

        }

        // If the board is full, it's a tie

        else if (isBoardFull(board)) {

            return 0;

        }

        // If the game is still ongoing

        return Integer.MAX\_VALUE;

    }

    // Minimax function to find the best move

    public static int minimax(char[][] board, int depth, boolean isMaximizing) {

        int score = evaluateBoard(board);

        // If the game is over, return the score

        if (score != Integer.MAX\_VALUE) {

            return score;

        }

        // If it's the AI's turn (maximizing player)

        if (isMaximizing) {

            int maxScore = Integer.MIN\_VALUE;

            for (int i = 0; i < 3; ++i) {

                for (int j = 0; j < 3; ++j) {

                    if (board[i][j] == ' ') {

                        board[i][j] = 'O';

                        maxScore = Math.max(maxScore, minimax(board, depth + 1, !isMaximizing));

                        board[i][j] = ' '; // Undo the move

                    }

                }

            }

            return maxScore;

        }

        // If it's the player's turn (minimizing player)

        else {

            int minScore = Integer.MAX\_VALUE;

            for (int i = 0; i < 3; ++i) {

                for (int j = 0; j < 3; ++j) {

                    if (board[i][j] == ' ') {

                        board[i][j] = 'X';

                        minScore = Math.min(minScore, minimax(board, depth + 1, !isMaximizing));

                        board[i][j] = ' '; // Undo the move

                    }

                }

            }

            return minScore;

        }

    }

    // Function to make the AI move using minimax

    public static void makeAIMove(char[][] board) {

        int bestScore = Integer.MIN\_VALUE;

        int bestMoveRow = -1;

        int bestMoveCol = -1;

        for (int i = 0; i < 3; ++i) {

            for (int j = 0; j < 3; ++j) {

                if (board[i][j] == ' ') {

                    board[i][j] = 'O';

                    int moveScore = minimax(board, 0, false);

                    board[i][j] = ' '; // Undo the move

                    if (moveScore > bestScore) {

                        bestScore = moveScore;

                        bestMoveRow = i;

                        bestMoveCol = j;

                    }

                }

            }

        }

        board[bestMoveRow][bestMoveCol] = 'O';

    }

    // Function to play the Tic-Tac-Toe game

    public static void playGame() {

        char[][] board = {

            {' ', ' ', ' '},

            {' ', ' ', ' '},

            {' ', ' ', ' '}

        };

        Scanner scanner = new Scanner(System.in);

        System.out.print("Do you want to play first? (y/n): ");

        char choice = scanner.next().charAt(0);

        int currentPlayer;

        if (choice == 'y' || choice == 'Y') {

            currentPlayer = 1; // Player starts first

        } else {

            currentPlayer = 2; // AI starts first

        }

        while (true) {

            System.out.println("Current Board:");

            printBoard(board);

            if (currentPlayer == 1) {

                // Player's turn

                int row, col;

                System.out.print("Enter your move (row and column): ");

                row = scanner.nextInt();

                col = scanner.nextInt();

                if (row < 0 || row >= 3 || col < 0 || col >= 3 || board[row][col] != ' ') {

                    System.out.println("Invalid move! Try again.");

                    continue;

                }

                board[row][col] = 'X';

            } else {

                // AI's turn

                makeAIMove(board);

            }

            // Check if the current player has won

            if (checkWin(board, (currentPlayer == 1) ? 'X' : 'O')) {

                System.out.println("Current Board:");

                printBoard(board);

                System.out.println("Player " + currentPlayer + " wins!");

                break;

            }

            // Check if the game is a tie

            if (isBoardFull(board)) {

                System.out.println("Current Board:");

                printBoard(board);

                System.out.println("It's a tie!");

                break;

            }

            // Switch to the next player

            currentPlayer = 3 - currentPlayer;

        }

        scanner.close();

    }

}