Name – Pranav Salunkhe

TY CS-D

Batch-1

Roll no – 09

Assignment 1 B

Code –

class XOGameAI {

    static class Move {

        int row, col;

    }

    static char player = 'x', opponent = 'o';

    static Boolean movesLeft(char board[][]) {

        for (int i = 0; i < 3; i++) {

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

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

                    return true;

            }

        }

        return false;

    }

    static int evaluate(char b[][]) {

        for (int i = 0; i < 3; i++) {

            if (b[i][0] == b[i][1] && b[i][1] == b[i][2]) {

                if (b[i][0] == player)

                    return +10;

                else

                    return -10;

            }

        }

        for (int i = 0; i < 3; i++) {

            if (b[0][i] == b[1][i] && b[1][i] == b[2][i]) {

                if (b[0][i] == player)

                    return +10;

                else

                    return -10;

            }

        }

        if (b[0][0] == b[1][1] && b[1][1] == b[2][2]) {

            if (b[0][0] == player)

                return +10;

            else if (b[0][0] == opponent)

                return -10;

        }

        if (b[0][2] == b[1][1] && b[1][1] == b[2][0]) {

            if (b[0][2] == player)

                return +10;

            else if (b[0][2] == opponent)

                return -10;

        }

        return 0;

    }

    static int minimax(char board[][], int depth, Boolean isMax) {

        int score = evaluate(board);

        if (score == 10 || score == -10)

            return score;

        if (movesLeft(board))

            return 0;

        if (isMax) {

            int best = -1000;

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

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

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

                        board[i][j] = player;

                        best = Math.max(best, minimax(board, depth + 1, !isMax));

                        board[i][j] = '\_';

                    }

                }

            }

            return best;

        } else {

            int best = 1000;

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

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

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

                        board[i][j] = player;

                        best = Math.max(best, minimax(board, depth + 1, !isMax));

                        board[i][j] = '\_';

                    }

                }

            }

            return best;

        }

    }

    static Move findBestMove(char board[][]) {

        int bestVal = -1000;

        Move bestMov = new Move();

        bestMov.row = -1;

        bestMov.col = -1;

        for (int i = 0; i < 3; i++) {

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

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

                    board[i][j] = player;

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

                    board[i][j] = '\_';

                    if (moveVal > bestVal) {

                        bestMov.row = i;

                        bestMov.col = j;

                        bestVal = moveVal;

                    }

                }

            }

        }

        System.out.println("Best move " + bestVal);

        return bestMov;

    }

    public static void main(String[] args) {

        System.out.println(Integer.MIN\_VALUE);

        char board[][] = {

                { 'o', 'o', 'x' },

                { 'o', 'o', 'x' },

                { '\_', '\_', '\_' } };

        Move bestMove = findBestMove(board);

        System.out.printf("The Optimal Move is :\n");

        System.out.printf("ROW: %d COL: %d\n\n",

                bestMove.row, bestMove.col);

    }

}

Output

