**VISHWAKARMA INSTITUTE OF TECHNOLOGY**

COMPUTER ENGINEERING

**Name: Abhinav Mahajan**

**Division: TY-C**

**Roll No: 15**

**Subject: Artificial Intelligence (AI)**

**LAB ASSIGNMENT NO – 1**

Implementation of AI and Non-AI techniques by implementing Tic-Tac-Toe Game.

1. **Non-AI Technique:**

*#include* <bits/stdc++.h>

using namespace std;

vector<vector<char>> board(3, vector<char>(3, ' '));

void displayBoard(){

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

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

            cout<<board[i][j]<<" ";

        }

        cout<<endl;

    }

}

bool checkWin(char symbol) {

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

*if* (board[i][0] == symbol && board[i][1] == symbol && board[i][2] == symbol)

*return* true;

*if* (board[0][i] == symbol && board[1][i] == symbol && board[2][i] == symbol)

*return* true;

    }

*if* (board[0][0] == symbol && board[1][1] == symbol && board[2][2] == symbol)

*return* true;

*if* (board[0][2] == symbol && board[1][1] == symbol && board[2][0] == symbol)

*return* true;

*return* false;

}

bool makeMove(int row, int col, char symbol) {

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

*return* false;

    board[row][col] = symbol;

*return* true;

}

bool isBoardFull(){

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

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

*if*(board[i][j] == ' ') *return* false;

        }

    }

*return* true;

}

void makeComputerMove() {

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

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

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

                board[i][j] = 'O';

*if* (checkWin('O'))

*return*;

                board[i][j] = ' ';

            }

        }

    }

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

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

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

                board[i][j] = 'X';

*if* (checkWin('X')) {

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

*return*;

                }

                board[i][j] = ' ';

            }

        }

    }

    int row, col;

*while* (!makeMove(row, col, 'O')) {

        row = rand() % 3;

        col = rand() % 3;

    }

}

int main() {

    char currentPlayer = 'X';

    int row, col;

*while* (true) {

        displayBoard();

*if* (currentPlayer == 'X') {

            cout << "Player X's turn. Enter row and column: ";

            cin >> row >> col;

*if* (!makeMove(row, col, currentPlayer)) {

                cout << "Invalid move. Try again." << endl;

*continue*;

            }

        } *else* {

            cout << "Player O's turn" << endl;

            makeComputerMove();

        }

*if* (checkWin(currentPlayer)) {

            displayBoard();

            cout << "Player " << currentPlayer << " wins!" << endl;

*break*;

        }

*if* (isBoardFull()) {

            displayBoard();

            cout << "It's a draw!" << endl;

*break*;

        }

        currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';

    }

*return* 0;

}

1. **AI Technique:**

*#include*<bits/stdc++.h>

using namespace std;

int BOARD\_SIZE = 3;

char PLAYER\_X = 'X';

char PLAYER\_O = 'O';

char EMPTY\_CELL = ' ';

void printBoard(vector<vector<char>>& board){

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

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

            cout<<board[i][j]<<" ";

        }

        cout<<endl;

    }

}

bool checkWin(vector<vector<char>>& board, char player) {

*for* (int i = 0; i < BOARD\_SIZE; i++) {

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

*return* true;

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

*return* true;

    }

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

*return* true;

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

*return* true;

*return* false;

}

bool isBoardFull(vector<vector<char>>& board) {

*for* (int i = 0; i < BOARD\_SIZE; i++) {

*for* (int j = 0; j < BOARD\_SIZE; j++) {

*if* (board[i][j] == EMPTY\_CELL)

*return* false;

        }

    }

*return* true;

}

int evaluateBoard(vector<vector<char>>& board) {

*if* (checkWin(board, PLAYER\_X))

*return* 10;

*if* (checkWin(board, PLAYER\_O))

*return* -10;

*return* 0;

}

int minimax(vector<vector<char>>& board, int depth, bool isMaximizer) {

    int score = evaluateBoard(board);

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

*return* score;

*if* (isBoardFull(board))

*return* 0;

*if* (isMaximizer) {

        int bestScore = -1000;

*for* (int i = 0; i < BOARD\_SIZE; i++) {

*for* (int j = 0; j < BOARD\_SIZE; j++) {

*if* (board[i][j] == EMPTY\_CELL) {

                    board[i][j] = PLAYER\_X;

                    bestScore = max(bestScore, minimax(board, depth + 1, !isMaximizer));

                    board[i][j] = EMPTY\_CELL;

                }

            }

        }

*return* bestScore;

    } *else* {

        int bestScore = 1000;

*for* (int i = 0; i < BOARD\_SIZE; i++) {

*for* (int j = 0; j < BOARD\_SIZE; j++) {

*if* (board[i][j] == EMPTY\_CELL) {

                    board[i][j] = PLAYER\_O;

                    bestScore = min(bestScore, minimax(board, depth + 1, !isMaximizer));

                    board[i][j] = EMPTY\_CELL;

                }

            }

        }

*return* bestScore;

    }

}

pair<int, int> findBestMove(vector<vector<char>>& board) {

    int bestScore = -1000;

    pair<int, int> bestMove = {-1, -1};

*for* (int i = 0; i < BOARD\_SIZE; i++) {

*for* (int j = 0; j < BOARD\_SIZE; j++) {

*if* (board[i][j] == EMPTY\_CELL) {

                board[i][j] = PLAYER\_X;

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

                board[i][j] = EMPTY\_CELL;

*if* (moveScore > bestScore) {

                    bestScore = moveScore;

                    bestMove = {i, j};

                }

            }

        }

    }

*return* bestMove;

}

int main() {

    vector<vector<char>> board(BOARD\_SIZE, vector<char>(BOARD\_SIZE, EMPTY\_CELL));

    bool isPlayerXTurn = true;

*while* (!checkWin(board, PLAYER\_X) && !checkWin(board, PLAYER\_O) && !isBoardFull(board)) {

        printBoard(board);

*if* (isPlayerXTurn) {

            int row, col;

            cout << "Enter row (0-2): ";

            cin >> row;

            cout << "Enter column (0-2): ";

            cin >> col;

*if* (row >= 0 && row < BOARD\_SIZE && col >= 0 && col < BOARD\_SIZE && board[row][col] == EMPTY\_CELL) {

                board[row][col] = PLAYER\_O;

                isPlayerXTurn = false;

            } *else* {

                cout << "Invalid move. Try again!" << endl;

            }

        } *else* {

            cout << "AI is making a move..." << endl;

            pair<int, int> aiMove = findBestMove(board);

            board[aiMove.first][aiMove.second] = PLAYER\_X;

            isPlayerXTurn = true;

        }

    }

    printBoard(board);

*if* (checkWin(board, PLAYER\_X))

        cout << "AI wins!" << endl;

*else* *if* (checkWin(board, PLAYER\_O))

        cout << "You win!" << endl;

*else*

        cout << "It's a draw!" << endl;

*return* 0;

}