**Assignment No:1**

**Student Name: Rahul Yargop**

**Roll No: 65**

**Class: AI-C**

**PRN: 12210096**

Create a tic tak toe solving bot using ai and non ai techniques.

**Non AI Algorithm Breakdown:**

**1. Initialization**

* The board is a **3x3** matrix initialized with empty spaces (' ').
* The player chooses their symbol (X or O), and the CPU is assigned the opposite symbol.
* The variable counter keeps track of the number of moves made.

**2. Game Loop**

* The game continues until a **winner** is found or the **board is full**.
* The **current player** makes a move:
  + If it is the **human player**, they manually input their move.
  + If it is the **CPU**, it executes a strategic decision-making process.

**3. Player's Move (playermove)**

* The player enters a **row** and **column** number to place their symbol.
* If the chosen cell is occupied, they are asked to retry.

**4. Printing the Board (printboard)**

* Displays the board with formatting.

**5. Checking for a Winner (checkWinner)**

* Checks all **rows**, **columns**, and **diagonals** for three matching symbols.
* Returns 1 if a player has won; otherwise, it continues.

**6. CPU's Move (cpumove)**

* The CPU makes its move based on the **turn number** (counter):
  + **First Move (Turn 1):** Always places in the center if available.
  + **Second Move (Turn 2):** Uses the check() function to pick a strong position.
  + **Third Move and Beyond:**
    - First checks if it can **win in one move** (posswin).
    - If not, it checks if the **player is about to win** and blocks them.
    - If neither is possible, it picks a **strategic position**.

**7. CPU Decision-Making (check and posswin)**

* check(board, cpu): Selects a strategic empty position.
* posswin(board, player, res): Detects potential **winning moves** by checking row, column, and diagonal patterns.
* value(a): Assigns numeric values (X = 3, O = 5, empty = 2) to analyze board positions.

**8. Game End Conditions**

* If a **winner is found**, the game announces the result.
* If the **board is full** with no winner, it declares a **draw**.

**Key Features:**

1. **Basic AI Strategy**
   * Tries to **win in one move** if possible.
   * Blocks **player's winning moves**.
   * Plays in a **strategic empty spot** otherwise.
2. **Turn-based Execution**
   * Alternates between **player and CPU**.
3. **Input Validation**
   * Ensures players **don't overwrite existing moves**.

**CODE:**

#include<stdio.h>

#define SIZE 3

int counter =1;

void initializeBoard(char board[SIZE][SIZE]);

int checkWinner(char board[SIZE][SIZE]);

void playermove(char board[SIZE][SIZE], char player);

void printboard(char board[SIZE][SIZE]);

int isFull(char board[SIZE][SIZE]);

void cpumove(char board[SIZE][SIZE], char cpu, char player, int counter);

void check(char board[SIZE][SIZE], char cpu);

int posswin(char board[SIZE][SIZE], char player, int res[]);

int value(char a);

void initializeBoard(char board[SIZE][SIZE]){

    for(int i=0;i<SIZE;i++){

        for(int j=0;j<SIZE;j++){

            board[i][j] = ' ';

        }

    }

}

int checkWinner(char board[SIZE][SIZE]){

    for(int i = 0;i<SIZE;i++){

        if(board[i][0] != ' '&& board[i][0]==board[i][1] && board[i][1]==board[i][2]){

            return 1;

        }

        if(board[0][i] != ' '&& board[0][i]==board[1][i] && board[1][i]==board[2][i]){

            return 1;

        }

    }

    if (board[0][0] != ' ' && board[0][0] == board[1][1] && board[1][1] == board[2][2]) {

        return 1;

    }

    if (board[0][2] != ' ' && board[0][2] == board[1][1] && board[1][1] == board[2][0]) {

        return 1;

    }

    return 0;

}

void playermove(char board[SIZE][SIZE],char player){

    int row,col;

    printf("Player %c, enter your move (row and column): ", player);

    scanf("%d %d", &row, &col);

    while(1){

    if (row >= 0 && row < SIZE && col >= 0 && col < SIZE && board[row][col] == ' ') {

            board[row][col] = player;

            break;

        } else {

            printf("Invalid move. Try again.\n");

        }

    }

}

void printboard(char board[SIZE][SIZE]) {

    printf("\n");

    for (int i = 0; i < SIZE; i++) {

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

            printf(" %c ", board[i][j]);

            if (j < SIZE - 1) printf("|");

        }

        printf("\n");

        if (i < SIZE - 1) printf("---|---|---\n");

    }

    printf("\n");

}

int isFull( char board[SIZE][SIZE]) {

    for(int i=0;i<SIZE;i++){

        for(int j=0;j<SIZE;j++){

            if(board[i][j] == ' '){

                return 0;

            }

        }

    }

    return 1;

}

void cpumove(char board[SIZE][SIZE],char cpu,char player,int counter){

    int res[2];

    switch (counter)

    {

    case 1:

        board[1][1] = cpu;

        break;

    case 2:

        check(board, cpu);

        break;

    case 3:

        if(posswin(board, player,res)!=0){

            int a = res[0];

            int b = res[1];

            board[a][b] = cpu;

        }

        else

        check(board, cpu);

        break;

    case 4:

        if(posswin(board, cpu,res)!=0){

            int a = res[0];

            int b = res[1];

            board[a][b] = cpu;

        }

        else if(posswin(board, player,res)!=0){

            int a = res[0];

            int b = res[1];

            board[a][b] = cpu;

        }

        else

        check(board, cpu);

        break;

    case 5:

        if(posswin(board, cpu,res)!=0){

            int a = res[0];

            int b = res[1];

            board[a][b] = cpu;

        }

        else if(posswin(board, player,res)!=0){

            int a = res[0];

            int b = res[1];

            board[a][b] = cpu;

        }else

        check(board, cpu);

        break;

    case 6:

        if(posswin(board, cpu,res)!=0){

            int a = res[0];

            int b = res[1];

            board[a][b] = cpu;

        }

        else if(posswin(board, player,res)!=0){

            int a = res[0];

            int b = res[1];

            board[a][b] = cpu;

        }else

        check(board, cpu);

        break;

    case 7:

        if(posswin(board, cpu,res)!=0){

            int a = res[0];

            int b = res[1];

            board[a][b] = cpu;

        }

        else if(posswin(board, player,res)!=0){

            int a = res[0];

            int b = res[1];

            board[a][b] = cpu;

        }else

        check(board, cpu);

        break;

    case 8:

        if(posswin(board, cpu,res)!=0){

            int a = res[0];

            int b = res[1];

            board[a][b] = cpu;

        }

        else if(posswin(board, player,res)!=0){

            int a = res[0];

            int b = res[1];

            board[a][b] = cpu;

        }else

        check(board, cpu);

        break;

    case 9:

        if(posswin(board, cpu,res)!=0){

            int a = res[0];

            int b = res[1];

            board[a][b] = cpu;

        }

        else if(posswin(board, player,res)!=0){

            int a = res[0];

            int b = res[1];

            board[a][b] = cpu;

        }else

        check(board, cpu);

        break;

    default:

        break;

    }

}

void check(char board[SIZE][SIZE], char cpu){

    if(board[SIZE/2][SIZE/2]==' '){

            board[SIZE/2][SIZE/2]= cpu;

            return;

        }

        if(board[0][0]==' '){

            board[0][0]= cpu;

            return;

        }

        if(board[0][2]==' '){

            board[0][2]= cpu;

            return;

        }

        if(board[2][0]==' '){

            board[2][0]= cpu;

            return;

        }

        if(board[2][2]==' '){

            board[2][2]= cpu;

            return;

        }

        if(board[0][1]==' '){

            board[0][1]= cpu;

            return;

        }

        if(board[1][2]==' '){

            board[1][2]= cpu;

            return;

        }

        if(board[2][1]==' '){

           board[2][1]= cpu;

            return;

        }

        if(board[0][1]==' '){

            board[0][1]= cpu;

            return;

        }

}

int posswin(char board[SIZE][SIZE],char player,int res[]){

    for(int i=0;i<SIZE;i++){

        if(value(board[i][0])\*value(board[i][1])\*value(board[i][2])==18 ||value(board[i][0])\*value(board[i][1])\*value(board[i][2])==50){

            res[0] = i;

            for(int j=0;j<3;j++){

                if(board[i][j]==' '){

                    res[1] = j;

                    return 1;

                }

            }

        }

        if(value(board[0][i])\*value(board[1][i])\*value(board[2][i])==18 ||value(board[0][i])\*value(board[1][i])\*value(board[2][i])==50){

            res[1] = i;

            for(int j=0;j<3;j++){

                if(board[j][i]==' '){

                    res[0] = j;

                    return 1;

                }

            }

        }

    }

    if(value(board[0][0])\*value(board[1][1])\*value(board[2][2])==18 ||value(board[0][0])\*value(board[1][1])\*value(board[2][2])==50){

        for(int i=0;i<3;i++){

            for(int j=0;j<3;j++){

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

                    res[0] = i;

                    res[1] = i;

                    return 1;

                }

            }

        }

    }

    if(value(board[2][0])\*value(board[1][1])\*value(board[0][2])==18 ||value(board[2][0])\*value(board[1][1])\*value(board[0][2])==50){

        if(board[2][0]==' '){

            res[0] = 2;

            res[1] = 0;

        }

        if(board[1][1]==' '){

            res[0] = 1;

            res[1] = 1;

        }

        if(board[0][2]==' '){

            res[0] = 0;

            res[1] = 2;

        }

    }

    return 0;

}

int value(char a){

    if(a == ' ') return 2;

    if(a == 'X') return 3;

    if(a=='O' ) return 5;

}

int main(){

    char board[SIZE][SIZE];

    char player,cpu,currentPlayer = 'X';

    int winner = 0;

    printf("Choose your Symbol (X or O)");

    scanf("%c", &player);

    cpu = (player=='X')? 'O':'X';

    initializeBoard(board);

    while(!winner && !isFull(board)){

        printboard(board);

        if(currentPlayer == player){

            playermove(board,currentPlayer);

            counter++;

        }else{

            cpumove(board,cpu,player,counter);

            counter++;

        }

        winner = checkWinner(board);

        if(winner==0){

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

        }

    }

    printboard(board);

    if (winner) {

        printf("Player %c wins!\n", currentPlayer);

    } else {

        printf("It's a draw!\n");

    }

    return 0;

}

**OUTPUT:**

E:\Third Year\Sem 2\Aritificial Intelligence\LAB\Assignment 1>cd "e:\Third Year\Sem 2\Aritificial Intelligence\LAB\Assignment 1\" && gcc tic\_tqak\_toe\_nonai2.c -o tic\_tqak\_toe\_nonai2 && "e:\Third Year\Sem 2\Aritificial Intelligence\LAB\Assignment 1\"tic\_tqak\_toe\_nonai2

Choose your Symbol (X or O)X

   |   |

---|---|---

   |   |

---|---|---

   |   |

Player X, enter your move (row and column): 1 1

   |   |

---|---|---

   | X |

---|---|---

   |   |

 O |   |

---|---|---

   | X |

---|---|---

   |   |

Player X, enter your move (row and column): 0 2

 O |   | X

---|---|---

   | X |

---|---|---

   |   |

 O |   | X

---|---|---

   | X |

---|---|---

 O |   |

Player X, enter your move (row and column): 1 0

 O |   | X

---|---|---

 X | X |

---|---|---

 O |   |

 O |   | X

---|---|---

 X | X | O

---|---|---

 O |   |

Player X, enter your move (row and column): 2 1

 O |   | X

---|---|---

 X | X | O

---|---|---

 O | X |

 O | O | X

---|---|---

 X | X | O

---|---|---

 O | X |

Player X, enter your move (row and column): 2 2

 O | O | X

---|---|---

 X | X | O

---|---|---

 O | X | X

It's a draw!