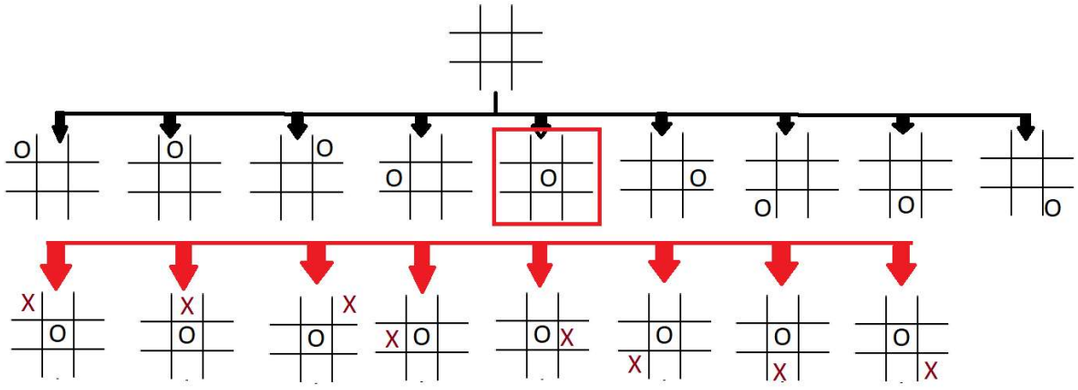
Task 2: Implementation of Tic-Tac-Toe Game:

Task 2.1:



Task 2.2:

Video link:
https://youtu.be/q8OljUgrdRk

MiniMax Code:

```
//Muhammad Somaan 2528404
//Muhammed Hashir Faraz 2528545

#include <stdio.h>
#include <time.h>
#include <ctype.h>
#include <malloc.h>

#define SIZE 3

struct TicTacToe
{
    char board[SIZE][SIZE];
    char playerChar;
    char AIChar;
};
typedef struct TicTacToe TicTacToe;

struct Move
{
    int x;
    int y;
};
typedef struct Move Move;


TicTacToe *createGame();
char getPlayerChar();

void playPlayerTurn(TicTacToe *game);

void playAITurn(TicTacToe* game);

Move* minimax(TicTacToe* game);
int max(TicTacToe* game, Move *move);
```

```c
int min(TicTacToe* game, Move *move);

int checkWin(TicTacToe* game, char player);

int isBoardFull(TicTacToe* game);
int evaluateBoard(TicTacToe* game);

void printBoard(TicTacToe *game);


int checkBoardStatus(TicTacToe *game);

int main()
{
    TicTacToe *game = createGame();
    int continueGame = 1;
    printBoard(game);

    while(continueGame)
    {
        playPlayerTurn(game);
        printBoard(game);
        continueGame = checkBoardStatus(game);

        if(!continueGame)
            break;

        clock_t start = clock();
        // Call the Minimax algorithm to calculate the next move
        playAITurn(game);

        clock_t end = clock();
        double elapsed_time = (double)(end - start) / CLOCKS_PER_SEC;
        printf("Elapsed time: %f\n", elapsed_time);

        printBoard(game);
        continueGame = checkBoardStatus(game);
    }

    printBoard(game);

    return 0;
}



TicTacToe *createGame()
{
    TicTacToe *game = (TicTacToe *) malloc(sizeof(TicTacToe));

    int i;
    int j;

    for(i=0; i<SIZE; i++)
        for(j=0; j<SIZE; j++)
            game->board[i][j] = '_';

    game->playerChar = getPlayerChar();
    game->AIChar = (game->playerChar == 'X') ? 'O' : 'X';

    return game;
```

```c
}

char getPlayerChar()
{
    char playerChar = ' ';

    while (playerChar != 'X' && playerChar != 'O')
    {
        printf("Select your character X or O: ");
        scanf(" %c", &playerChar);
        playerChar = toupper(playerChar);
    }

    printf("You selected: %c\n", playerChar);
    return playerChar;
}

void playPlayerTurn(TicTacToe *game)
{
    int row = -1;
    int col = -1;

    while (row < 0 || row >= SIZE || col < 0 || col >= SIZE || game-
>board[row][col] != '_')
    {
        printf("Enter the row and column for your move with no overlap eg:
(0 0): ");
        scanf(" %d %d", &row, &col);
    }

    game->board[row][col] = game->playerChar;
}


void playAITurn(TicTacToe* game)
{
    Move *move = minimax(game);
    game->board[move->x][move->y] = game->AIChar;
}


Move* minimax(TicTacToe* game)
{
    Move *move = (Move *) malloc(sizeof(Move));
    max(game, move);

    return move;
}

int max(TicTacToe* game, Move *move)
{
    if (checkWin(game, game->playerChar))
        return -1; // Player wins
    else if (checkWin(game, game->AIChar))
        return 1;
    else if (isBoardFull(game))
        return 0;

    int maxScore = INT_MIN;
    int bestMoveX = -1;
    int bestMoveY = -1;
```

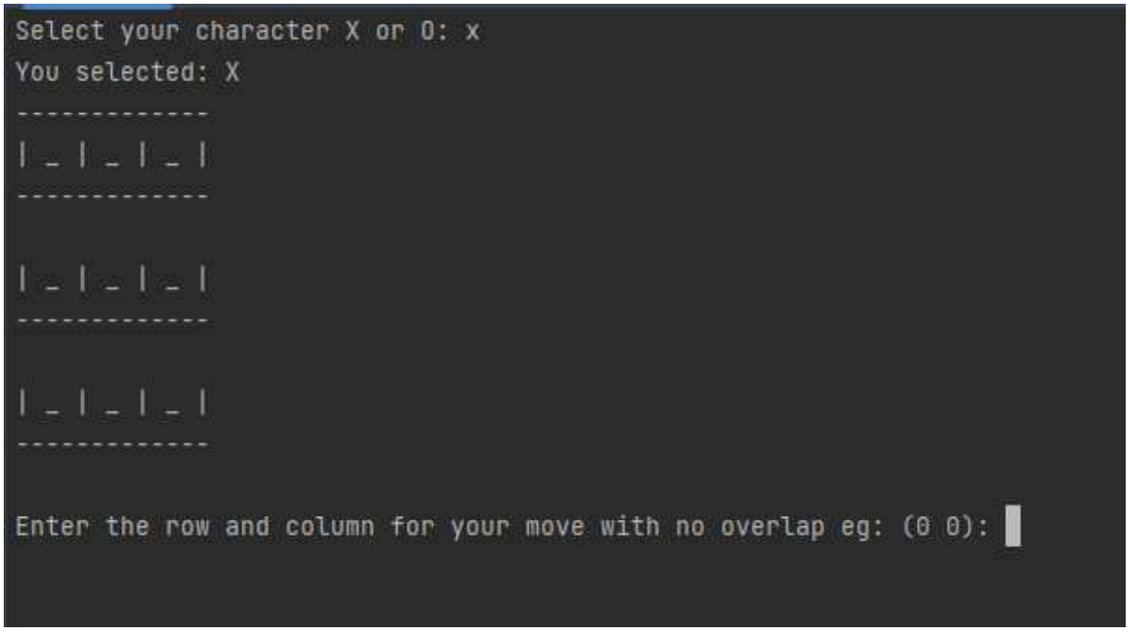
```
    for (int i = 0; i < SIZE; i++)
    {
        for (int j = 0; j < SIZE; j++)
        {
            if (game->board[i][j] == '_')
            {
                game->board[i][j] = game->AIChar;
                int currentScore = min(game, move);

                //reverting the change back, so our original board is not
changed.
                game->board[i][j] = '_';

                if (currentScore > maxScore)
                {
                    bestMoveX = i;
                    bestMoveY = j;
                    maxScore = currentScore;
                }
            }
        }
    }

    move->x = bestMoveX;
    move->y = bestMoveY;

    return maxScore;
}

int min(TicTacToe* game, Move *move)
{
    if (checkWin(game, game->playerChar))
        return -1; // Player wins
    else if (checkWin(game, game->AIChar))
        return 1;
    else if (isBoardFull(game))
        return 0;

    int minScore = INT_MAX;

    for (int i = 0; i < SIZE; i++)
    {
        for (int j = 0; j < SIZE; j++)
        {
            if (game->board[i][j] == '_')
            {
                game->board[i][j] = game->playerChar;
                int currentScore = max(game, move);
                game->board[i][j] = '_';

                if (currentScore < minScore)
                    minScore = currentScore;
            }
        }
    }

    return minScore;
}
```

```c
int checkWin(TicTacToe* game, char player)
{

    //Checks board horizontally
    for (int x = 0; x < SIZE; x++)
    {
        int horizontal = 1;

        for (int y = 0; y < SIZE; y++)
            horizontal = horizontal && player == game->board[x][y];

        if (horizontal)
            return 1;
    }

    //Checks board vertically
    for (int y = 0; y < SIZE; y++)
    {
        int vertical = 1;

        for (int x = 0; x < SIZE; x++)
            vertical = vertical && player == game->board[x][y];

        if (vertical)
            return 1;
    }

    //Checks board diagonally
    int diagonal = 1;
    int diagonal1 = 1;


    for (int x = 0; x < SIZE; x++)
        diagonal = diagonal && player == game->board[x][x];
    for (int x = 0, y = SIZE - 1; x < SIZE; x++, y--)
        diagonal1 = diagonal1 && player == game->board[x][y];
    if (diagonal || diagonal1)
        return 1;


    return 0;
}


int isBoardFull(TicTacToe* game)
{
    int i;
    int j;

    for (i = 0; i < SIZE; i++)
        for (j = 0; j < SIZE; j++)
            if (game->board[i][j] == '_')
                return 0;
    return 1;
}


int evaluateBoard(TicTacToe* game)
{
    if (checkWin(game, game->playerChar))
        return -1; // Player wins
```

```
    else if (checkWin(game, game->AIChar))
        return 1; // AI wins
    else
        return 0; // Draw
}

int checkBoardStatus(TicTacToe *game)
{
    if(isBoardFull(game))
    {
        printf("Its a draw!\n");
        return 0;
    }
    else
    {
        switch(evaluateBoard(game))
        {
            case -1:
                printf("You won!\n");
                return 0;
            case 1:
                printf("You lost!\n");
                return 0;
        }
    }
    return 1;
}


void printBoard(TicTacToe *game)
{
    printf("-------------\n");
    for (int i = 0; i < SIZE; i++)
    {
        for (int j = 0; j < SIZE; j++)
            printf("| %c ", game->board[i][j]);

        printf("|\n");
        printf("-------------\n\n");
    }
}
```

Outputs:

Tic-tac-toe board:



Draw:

AI wins:


Elapsed time: 0.000000
X	X	O
X	O	_
O	_	_
You lost!
X	X	O
X	O	_
O	_	_
Process finished with exit code 0