

Annexure-I

Title of the work

Name of the Organization/ Company/Project

A training report

Submitted in partial fulfillment of the requirements for the award of degree of

B.Tech CSE

Machine Learning

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB



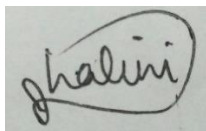
From 06/28/24 to 08/15/24

SUBMITTED BY

Name of student: Shalini Tigga

Registration Number: 12205663

Signature of the student :

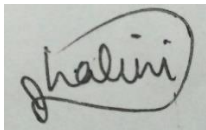
A handwritten signature in black ink, appearing to read 'shalini', is shown within a rectangular box.

Annexure – II: Student Declaration
To whom so ever it may concern

I, **Shalini Tigga, 12205663,** hereby declare that the work done by me on “**Tic- Tac- Toe Game**” from **June, year** to **August, Year,** is a record of original work for the partial fulfillment of the requirements for the award of the degree, **B.Tech CSE.**

Shalini Tigga (12205663)

Signature of the student

A handwritten signature in black ink, appearing to read 'shalini', enclosed within a hand-drawn oval border.

Dated: 08/30/2024

Training Certification from organization



CERTIFICATE

OF COURSE COMPLETION

THIS IS TO CERTIFY THAT

Shalini Tigga

has successfully completed a 26-week course on Complete Interview Preparation.

Sandeep Jain

Mr. Sandeep Jain

Founder & CEO, GeeksforGeeks

<https://media.geeksforgeeks.org/courses/certificates/de04a2de1f3eca1066b777141b99208a.pdf>

<https://media.geeksforgeeks.org/courses/certificates/de04a2de1f3eca1066b777141b99208a.pdf>

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Chirag Vaishnav, whose guidance and support were instrumental in the successful completion of this Tic-Tac-Toe game implementation in C++. Their valuable insights and feedback helped me navigate the complexities of game development and sharpen my programming skills.

I also wish to extend my appreciation to my peers and classmates, whose collaborative spirit and willingness to share ideas fostered a productive learning environment. Their encouragement and constructive criticism were crucial in refining my approach to this project.

Lastly, I would like to thank my family and friends for their unwavering support and understanding throughout the course of this project. Their belief in my abilities provided the motivation needed to see this project through to completion.

Thank you all for your contributions, without which this report would not have been possible.

TABLE OF CONTENTS

S.No	Topic	Page No.
1	Student Declaration	1
2	Certificate	2
3	Acknowledgement	3
4	Introduction of the project undertaken	5
5	Introduction of the company/work	6 – 8
6	Brief description of the work done	9
7	System requirement	10
8	Design and Implementation	11 - 12
9	Code Explanation	13 – 15
10	Testing and Result	16 – 24
11	Conclusion	25
12	References	26
13	Appendices	27 - 34

INTRODUCTION OF THE PROJECT UNDERTAKEN

Objectives of the work undertaken

- Create and develop a Tic-Tac-Toe game in C++ that runs on the console.
- To showcase how basic programming ideas like loops, conditionals, arrays, and functions are utilized.
- To create a basic, engaging game where two players can compete with each other.
- To implement best practices in software development, including modular coding, error handling, and user interface design.

Scope of the Work

The report details the entire progression of creating a Tic-Tac-Toe game, from initial concept to final execution, and explains the fundamental logic for identifying win, loss, or draw scenarios within the game.

It centers on creating the game for a player on a 3x3 board without advanced features like AI opponent, GUI, or network play.

Importance and Applicability

Importance:

- The project serves as an educational tool for learning basic programming concepts and applying them in a real-world scenario.
- It introduces fundamental game development principles that can be scaled to more complex projects.
- Encourages problem-solving and logical thinking, which are essential skills in software development.

Applicability:

- The Tic-Tac-Toe game can be used as a classroom example for teaching C++ to beginners.
- The project can serve as a foundation for further development into more complex games or adding features like AI.
- The skills learned through this project are applicable to various areas of software development, including game design, algorithm implementation, and software engineering.

Role and Profile

Role:

- **Lead Developer:** Responsible for the entire development process, including the design, coding, testing, and documentation of the Tic-Tac-Toe game.
- **Designer:** Created the game flow, user interface, and logic for detecting game states.
- **Tester:** Conducted various test cases to ensure the game functions correctly and handles all possible user inputs.

Profile:

- **Author:** A computer science student (or software developer) with a strong interest in programming and game development. This project is part of a broader study to improve coding skills in C++ and understand basic game mechanics.
- **Skill Set:** Proficient in C++ programming, problem-solving, and logical thinking. Experience with software development methodologies and project documentation.

INTRODUCTION OF THE COMPANY/WORK

Company's Vision and Mission

The mission of the company is to offer top-notch, easy-to-access learning materials for computer science and programming. Their extensive collection of articles, tutorials, coding challenges, and courses clearly shows their commitment to serving learners at all skill levels. Additionally, to enable individuals to thrive in their technical professional paths. GeeksforGeeks places a strong emphasis on preparing for interviews, assisting users in securing positions with leading tech firms. Moreover, in order to establish a robust community of technology enthusiasts. The contributor program and campus ambassador initiatives promote collaboration and knowledge sharing on the platform.

GeeksforGeeks strives to be the top source for all things related to computer science. Their vision is to be the primary choice for learners and professionals seeking trustworthy technical information. They seek to provide equal access to technical education by offering top-notch resources to all individuals, regardless of their background or location. They aim to promote innovation in the tech field by creating a community of dedicated learners that will inspire future tech leaders and innovators.

Origin and growth of company

GeeksforGeeks (GFG) is a computer science portal founded by Sandeep Jain in 2009. It started as a blog to share computer science concepts and has grown into a comprehensive platform for learning and practicing coding. It is based in Noida, Uttar Pradesh, India.

It offers a diverse selection of articles covering different topics in computer science and programming. Provides users with practice problems and coding competitions to enhance their coding abilities. Offers interview insights and study materials for individuals pursuing tech industry positions. Provides online classes and guides covering a range of technology subjects. It is known as one of the top computer science websites in India and has a vast global user base of millions.

Various Departments and their functions

1. Content Creation
Responsible for producing high-quality educational content, including articles, tutorials, and study guides on various computer science topics such as data structures, algorithms, programming languages, and interview preparation.
2. Video Content and Production
Handles the creation and production of video tutorials, webinars, and other visual learning resources. This includes scripting, recording, and editing videos.
3. Practice and Competitive Programming

Develops and maintains the coding practice platform, which includes coding challenges, contests, and mock interviews. They ensure the availability of a vast range of problems for users to practice and improve their coding skills.

4. Tech Development

Focuses on the development and maintenance of the GFG website and associated applications. They work on enhancing the user experience, implementing new features, and ensuring the platform is robust and scalable.

5. Sales and Marketing

Manages the promotion of GFG's products and services, handles partnerships and sponsorships, and drives revenue through various channels like course sales, advertisements, and corporate tie-ups.

6. Customer Support

Provides support to users, handling queries related to course content, subscriptions, technical issues, and more. They ensure a smooth and satisfactory experience for all users.

7. Course Development

Develops and curates online courses offered by GFG. This includes selecting topics, creating curriculum, and working with instructors to deliver comprehensive learning experiences.

8. Human Resources

Manages recruitment, employee relations, and organizational development. HR ensures the company has the right talent and that employees are motivated and well-supported.

9. Finance and Administration

Handles all financial aspects, including budgeting, accounting, and financial planning. This department also manages administrative functions, ensuring smooth day-to-day operations.

10. Legal and Compliance

Ensures that GFG operates within the legal frameworks and complies with regulations. They handle contracts, intellectual property issues, and any legal disputes.

BRIEF DESCRIPTION OF THE WORK DONE

What is Tic-Tac-Toe ?

Tic-tac-toe, noughts and crosses, or Xs and Os is a paper-and-pencil game for two players who take turns marking the spaces in a three-by-three grid with X or O. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner. It is a solved game, with a forced draw assuming best play from both players.

Games like this that were played using boards with three-in-a-row traces back to ancient Egypt, where game boards of this kind have been discovered on roofing tiles from approximately 1300 BC.

A version of tic-tac-toe was played in ancient Rome, approximately in the first century BC. It was known as terni lapilli (three pebbles at once) and instead of a varying number of pieces, each player only had three; therefore, they had to shift them to vacant spots to continue playing. Chalked grid markings from the game have been discovered scattered throughout Rome. Another similar ancient game is three men's morris, played on a basic grid and needing three pieces in a line to win, as well as Picaria, a game of the Pueblos.

In 1952, OXO (also known as Noughts and Crosses), created by British computer scientist Sandy Douglas specifically for the EDSAC computer at the University of Cambridge, was recognized as one of the earliest video games. The computer opponent was able to play flawless games of tic-tac-toe against a human player.

In the year 1975, MIT students utilized tic-tac-toe to showcase the computational capabilities of Tinkertoy components. The Tinkertoy computer, constructed mostly of Tinkertoys, can play tic-tac-toe flawlessly. It is currently exhibited at the Computer History Museum.

SYSTEM REQUIREMENT

Hardware Requirement:

- Processor: 1.8 GHz or faster processor, with a quad-core or better recommended
- RAM: 2 GB of RAM, but 8 GB is recommended
- Hard disk space: 800 MB minimum, but up to 210 GB of available space depending on features installed
- Video card: Supports a minimum display resolution of 720p (1280 by 720)
- Operating system: Microsoft Windows

Software Requirement:

- Compiler used : Online gdb
- Link : <https://www.onlinegdb.com>

DESIGN AND IMPLEMENTATION

Game flow

Game Initialization

- Start: Display a welcome message and game instructions.
- Initialize Board: Create a 3x3 board represented by a 2D array or vector initialized with empty spaces or numbers (1-9) to indicate available spots.
- Set Players: Assign symbols to players (e.g., Human as 'X' and Computer as 'O').

Main Game Loop

- Check Game Status:
 1. Before each move, check if the game is over (win, loss, or draw).
 2. If the game is over, exit the loop and proceed to the end-game sequence.
- Player Move:
 1. Prompt Player: Ask the user if they want to play first and then to select a position (1-9) to place their symbol.
 2. Input Validation:
 - Ensure the input is a valid number within the board's range.
 - Ensure the selected position is not already occupied.
 3. Update Board: Place the player's symbol on the board.
- Display Board:
 1. After each move, display the updated board to the players.
- Switch Players:
 1. Toggle the current player (if Human just played, switch to Computer, and vice versa).

Check for Win Condition

- Winning Condition:
 1. Check if the current player has three of their symbols in a row (horizontally, vertically, or diagonally).
- Draw Condition:
 1. Check if all positions are filled without any player winning (a draw).
 2. If a draw is detected, the game ends.

End-Game Sequence

- Announce Result:
 1. If a player wins, announce the winner.

2. If the game is a draw, display a draw message.
- Replay Option:
 1. Ask if the players want to play again.
 2. If yes, reset the board and return to the game initialization step.
 3. If no, end the game and display a goodbye message.

Game Termination

- End: Thank the players for playing and exit the program.

CODE EXPLANATION

Data Structures used

`char board[SIDE][SIDE]`

Key Functions and their usage

`showBoard(char board[][SIDE])`

- Purpose: This function displays the current state of the Tic-Tac-Toe board.
- Functionality:
It prints the board in a 3x3 grid format.

`showInstructions()`

- Purpose: Displays the instructions to the player on how to select a cell on the board.
- Functionality:
It shows a sample 3x3 grid with numbers (1-9) representing the positions the player can choose to place their symbol.

`initialise(char board[][SIDE])`

- Purpose: Initializes the board by setting all cells to the '*' character.
- **Functionality:**
It loops through each cell of the board and assigns the character '*', indicating that the cell is empty.

`declareWinner(int whoseTurn)`

- Purpose: Announces the winner of the game.
- Functionality:
It checks who the last player was and prints whether the COMPUTER or HUMAN won.

`rowCrossed(char board[][SIDE])`

- Purpose: Checks if any row on the board has been completely crossed (i.e., all cells in the row contain the same player's symbol).
- Functionality:
 1. It iterates through each row and checks if all elements in a row are the same and not '*'.
 2. Returns true if a row is crossed, otherwise returns false.

`columnCrossed(char board[][SIDE])`

- **Purpose:** Checks if any column on the board has been completely crossed.
- **Functionality:**
 1. It iterates through each column and checks if all elements in a column are the same and not '*'.
 2. Returns true if a column is crossed, otherwise returns false.

diagonalCrossed(char board[][SIDE])

- Purpose: Checks if any of the diagonals on the board have been completely crossed.
- Functionality:
 1. It checks both diagonals ([0][0], [1][1], [2][2] and [0][2], [1][1], [2][0]) to see if they contain the same symbol and are not '*'.
 2. Returns true if a diagonal is crossed, otherwise returns false.

gameOver(char board[][SIDE])

- Purpose: Checks if the game is over by determining if any row, column, or diagonal has been crossed.
- Functionality:
 1. It uses the functions rowCrossed, columnCrossed, and diagonalCrossed to determine if the game is over.
 2. Returns true if the game is over, otherwise returns false.

minimax(char board[][SIDE], int depth, bool isAI)

- Purpose: Implements the minimax algorithm to determine the best possible move for the AI or Human.
- Functionality:
 1. If the game is over, it returns a score based on who won (+10 for HUMAN, -10 for COMPUTER).
 2. If the game is not over, it recursively explores all possible moves, calculates their minimax values, and returns the best score.

bestMove(char board[][SIDE], int moveIndex)

- Purpose: Determines the best move for the AI by calling the minimax function.
- Functionality:
 1. It iterates over all possible moves, uses the minimax function to evaluate each move, and selects the one with the best score.
 2. Returns the position (index) of the best move.

playTTT(int whoseTurn)

- Purpose: Manages the entire gameplay between the AI and the Human.
- Functionality:
 1. Initializes the game board.
 2. Shows the instructions.
 3. Alternates between the AI and Human turns, updating the board and checking the game status after each move.
 4. If the game ends, it declares the winner or a draw.

main()

- Purpose: Acts as the entry point for the program and handles user interaction.
- Functionality:
 1. It starts by displaying the Tic-Tac-Toe game title.
 2. Asks the player if they want to start first or let the AI start.

3. Based on the player's choice, it calls `playTTT` with the appropriate starting player.
4. After each game, it asks the player if they want to play again or quit.

TESTING AND RESULT

Test case 1 : Player plays first and quits after 1st round.

Output 1 : It results in a draw.

```
----- Tic-Tac-Toe -----
Do you want to start first? (y/n): y
Choose a cell numbered from 1 to 9 as below and play

  1 | 2 | 3
  ---
  4 | 5 | 6
  ---
  7 | 8 | 9

You can insert in the following positions: 1 2 3 4 5 6 7 8 9
Enter the position: 1
HUMAN has put a X in cell 1

  X | * | *
  ---
  * | * | *
  ---
  * | * | *
COMPUTER has put a O in cell 5

  X | * | *
  ---
  * | O | *
  ---
  * | * | *
```

```
You can insert in the following positions: 2 3 4 6 7 8 9
Enter the position: 2
HUMAN has put a X in cell 2

  X | X | *
  ---
  * | O | *
  ---
  * | * | *
COMPUTER has put a O in cell 3

  X | X | O
  ---
  * | O | *
  ---
  * | * | *
```

You can insert in the following positions: 4 6 7 8 9

Enter the position: 7

HUMAN has put a X in cell 7

```
  X | X | O
  -----
  * | O | *
  -----
  X | * | *
```

COMPUTER has put a O in cell 4

```
  X | X | O
  -----
  O | O | *
  -----
  X | * | *
```

You can insert in the following positions: 6 8 9

Enter the position: 6

HUMAN has put a X in cell 6

```
  X | X | O
  -----
  O | O | X
  -----
  X | * | *
```

COMPUTER has put a O in cell 8

```
  X | X | O
  -----
  O | O | X
  -----
  X | O | *
```

You can insert in the following positions: 9

Enter the position: 9

HUMAN has put a X in cell 9

```
  X | X | O
  -----
  O | O | X
  -----
  X | O | X
```

It's a draw

Do you want to quit? (y/n): y

Thank you for playing <3

Test case 2 : The player plays second and quits at the 2nd round

Output : Computer won.

```
----- Tic-Tac-Toe -----
Do you want to start first? (y/n): n
Choose a cell numbered from 1 to 9 as below and play

      1 | 2 | 3
      -----
      4 | 5 | 6
      -----
      7 | 8 | 9
COMPUTER has put a O in cell 1

      O | * | *
      -----
      * | * | *
      -----
      * | * | *
```

```
You can insert in the following positions: 2 3 4 5 6 7 8 9
Enter the position: 2
HUMAN has put a X in cell 2

      O | X | *
      -----
      * | * | *
      -----
      * | * | *
COMPUTER has put a O in cell 4

      O | X | *
      -----
      O | * | *
      -----
      * | * | *
```

You can insert in the following positions: 3 5 6 7 8 9

Enter the position: 7

HUMAN has put a X in cell 7

```
  O | X | *
-----
  O | * | *
-----
  X | * | *
```

COMPUTER has put a O in cell 5

```
  O | X | *
-----
  O | O | *
-----
  X | * | *
```

You can insert in the following positions: 3 6 8 9

Enter the position: 6

HUMAN has put a X in cell 6

```
  O | X | *
-----
  O | O | X
-----
  X | * | *
```

COMPUTER has put a O in cell 9

```
  O | X | *
-----
  O | O | X
-----
  X | * | O
```

COMPUTER has won

Do you want to start first? (y/n): y

Choose a cell numbered from 1 to 9 as below and play

```
  1 | 2 | 3
  ---
  4 | 5 | 6
  ---
  7 | 8 | 9
```

You can insert in the following positions: 1 2 3 4 5 6 7 8 9

Enter the position: 1

HUMAN has put a X in cell 1

```
  X | * | *
  ---
  * | * | *
  ---
  * | * | *
```

COMPUTER has put a O in cell 5

```
  X | * | *
  ---
  * | O | *
  ---
  * | * | *
```

You can insert in the following positions: 2 3 4 6 7 8 9

Enter the position: 4

HUMAN has put a X in cell 4

```
  X | * | *
  ---
  X | O | *
  ---
  * | * | *
```

COMPUTER has put a O in cell 7

```
  X | * | *
  ---
  X | O | *
  ---
  O | * | *
```

You can insert in the following positions: 2 3 6 8 9

Enter the position: 6

HUMAN has put a X in cell 6

```
  X | * | *
  -----
  X | O | X
  -----
  O | * | *
```

COMPUTER has put a O in cell 2

```
  X | O | *
  -----
  X | O | X
  -----
  O | * | *
```

You can insert in the following positions: 3 8 9

Enter the position: 9

HUMAN has put a X in cell 9

```
  X | O | *
  -----
  X | O | X
  -----
  O | * | X
```

COMPUTER has put a O in cell 3

```
  X | O | O
  -----
  X | O | X
  -----
  O | * | X
```

COMPUTER has won

Do you want to quit? (y/n): y

Thank you for playing <3

Test case 3 : Player plays first but inputs a already occupied cell.

Output : It results in a draw

```
----- Tic-Tac-Toe -----
Do you want to start first? (y/n): y
Choose a cell numbered from 1 to 9 as below and play

  1 | 2 | 3
  ---
  4 | 5 | 6
  ---
  7 | 8 | 9

You can insert in the following positions: 1 2 3 4 5 6 7 8 9
Enter the position: 1
HUMAN has put a X in cell 1

  X | * | *
  ---
  * | * | *
  ---
  * | * | *
COMPUTER has put a O in cell 5

  X | * | *
  ---
  * | O | *
  ---
  * | * | *
```

```
You can insert in the following positions: 2 3 4 6 7 8 9
Enter the position: 1
Position is occupied, select any one place from the available places

You can insert in the following positions: 2 3 4 6 7 8 9
Enter the position: 2
HUMAN has put a X in cell 2

      X | X | *
      ---
      * | O | *
      ---
      * | * | *
COMPUTER has put a O in cell 3

      X | X | O
      ---
      * | O | *
      ---
      * | * | *
```

You can insert in the following positions: 4 6 7 8 9

Enter the position: 7

HUMAN has put a X in cell 7

```
  X | X | O
-----
  * | O | *
-----
  X | * | *
```

COMPUTER has put a O in cell 4

```
  X | X | O
-----
  O | O | *
-----
  X | * | *
```

You can insert in the following positions: 6 8 9

Enter the position: 6

HUMAN has put a X in cell 6

```
  X | X | O
-----
  O | O | X
-----
  X | * | *
```

COMPUTER has put a O in cell 8

```
  X | X | O
-----
  O | O | X
-----
  X | O | *
```



```
You can insert in the following positions: 9
Enter the position: 8
Position is occupied, select any one place from the available places

You can insert in the following positions: 9
Enter the position: 9
HUMAN has put a X in cell 9

      X | X | O
      -----
      O | O | X
      -----
      X | O | X

It's a draw

Do you want to quit? (y/n): y

Thank you for playing <3
```

CONCLUSION

To conclude, this Tic-Tac-Toe implementation effectively demonstrates the application of the minimax algorithm in building a computer opponent capable of playing optimally. The program includes essential functions that manage the game board, handle user input, and determine the winner by checking rows, columns, and diagonals. Through the use of recursion in the minimax function, the program evaluates possible game states and selects the optimal move for the computer. This ensures that the computer opponent always makes the best possible move, leading to a challenging and fair gameplay experience for the human player.

The program is designed to be user-friendly, with clear instructions and feedback throughout the game. The game logic is robust, handling various scenarios such as invalid inputs, occupied positions, and determining draws or wins. Overall, this implementation not only serves as an engaging game but also as an educational tool for understanding basic game AI concepts, particularly in decision-making and adversarial search algorithms.

REFERENCES

- [1] <https://www.geeksforgeeks.org/courses/complete-interview-preparation>
- [2] <https://en.wikipedia.org/wiki/Tic-tac-toe>

APPENDICES

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

using namespace std;

#define COMPUTER 1
#define HUMAN 2
#define SIDE 3
#define COMPUTERMOVE 'O'
#define HUMANMOVE 'X'

void showBoard(char board[][SIDE]) {
    printf("\t\t\t %c | %c | %c \n", board[0][0], board[0][1], board[0][2]);
    printf("\t\t\t-----\n");
    printf("\t\t\t %c | %c | %c \n", board[1][0], board[1][1], board[1][2]);
    printf("\t\t\t-----\n");
    printf("\t\t\t %c | %c | %c \n", board[2][0], board[2][1], board[2][2]);
}

void showInstructions() {
    printf("\nChoose a cell numbered from 1 to 9 as below and play\n\n");
    printf("\t\t\t 1 | 2 | 3 \n");
    printf("\t\t\t-----\n");
    printf("\t\t\t 4 | 5 | 6 \n");
    printf("\t\t\t-----\n");
    printf("\t\t\t 7 | 8 | 9 \n");
}

void initialise(char board[][SIDE]) {
    for (int i = 0; i < SIDE; i++) {
        for (int j = 0; j < SIDE; j++) {
            board[i][j] = '*';
        }
    }
}
```

```

    }
}
}

```

```

void declareWinner(int whoseTurn) {
    if (whoseTurn == COMPUTER) {
        printf("COMPUTER has won\n");
    } else {
        printf("HUMAN has won\n");
    }
}

```

```

bool rowCrossed(char board[][SIDE]) {
    for (int i = 0; i < SIDE; i++) {
        if (board[i][0] == board[i][1] && board[i][1] == board[i][2] && board[i][0] != '*') {
            return true;
        }
    }
    return false;
}

```

```

bool columnCrossed(char board[][SIDE]) {
    for (int i = 0; i < SIDE; i++) {
        if (board[0][i] == board[1][i] && board[1][i] == board[2][i] && board[0][i] != '*') {
            return true;
        }
    }
    return false;
}

```

```

bool diagonalCrossed(char board[][SIDE]) {
    if (board[0][0] == board[1][1] && board[1][1] == board[2][2] && board[0][0] != '*') {
        return true;
    }
}

```

```

    }

    if (board[0][2] == board[1][1] && board[1][1] == board[2][0] && board[0][2] != '*') {
        return true;
    }

    return false;
}

```

```

bool gameOver(char board[][SIDE]) {
    return (rowCrossed(board) || columnCrossed(board) || diagonalCrossed(board));
}

```

```

int minimax(char board[][SIDE], int depth, bool isAI) {
    int score = 0;
    int bestScore = 0;
    if (gameOver(board)) {
        if (isAI) {
            return -10;
        } else {
            return +10;
        }
    } else {
        if (depth < 9) {
            if (isAI) {
                bestScore = -999;
                for (int i = 0; i < SIDE; i++) {
                    for (int j = 0; j < SIDE; j++) {
                        if (board[i][j] == '*') {
                            board[i][j] = COMPUTERMOVE;
                            score = minimax(board, depth + 1, false);
                            board[i][j] = '*';
                            if (score > bestScore) {
                                bestScore = score;
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

        }
    }
}

return bestScore;
} else {
    bestScore = 999;
    for (int i = 0; i < SIDE; i++) {
        for (int j = 0; j < SIDE; j++) {
            if (board[i][j] == '*') {
                board[i][j] = HUMANMOVE;
                score = minimax(board, depth + 1, true);
                board[i][j] = '*';
                if (score < bestScore) {
                    bestScore = score;
                }
            }
        }
    }
    return bestScore;
}
} else {
    return 0;
}
}
}

```

```

int bestMove(char board[][SIDE], int moveIndex) {
    int x = -1, y = -1;
    int score = 0, bestScore = -999;
    for (int i = 0; i < SIDE; i++) {
        for (int j = 0; j < SIDE; j++) {
            if (board[i][j] == '*') {
                board[i][j] = COMPUTERMOVE;

```

```

        score = minimax(board, moveIndex + 1, false);

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

        if (score > bestScore) {

            bestScore = score;

            x = i;

            y = j;

        }

    }

}

return x * SIDE + y;

}

```

```

void playTTT(int whoseTurn) {

    char board[SIDE][SIDE];

    int moveIndex = 0, x = 0, y = 0;

    initialise(board);

    showInstructions();

    while (gameOver(board) == false && moveIndex != SIDE * SIDE) {

        int n;

        if (whoseTurn == COMPUTER) {

            n = bestMove(board, moveIndex);

            x = n / SIDE;

            y = n % SIDE;

            board[x][y] = COMPUTERMOVE;

            printf("COMPUTER has put a %c in cell %d\n\n", COMPUTERMOVE, n + 1);

            showBoard(board);

            moveIndex++;

            whoseTurn = HUMAN;

        } else if (whoseTurn == HUMAN) {

            printf("\nYou can insert in the following positions: ");

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

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

```



```

        if (board[i][j] == '*') {
            printf("%d ", (i * 3 + j) + 1);
        }
    }
}

printf("\n\nEnter the position: ");
scanf("%d", &n);

n--;

x = n / SIDE;
y = n % SIDE;

if (board[x][y] == '*' && n < 9 && n >= 0) {
    board[x][y] = HUMANMOVE;

    printf("\nHUMAN has put a %c in cell %d\n", HUMANMOVE, n + 1);

    showBoard(board);

    moveIndex++;

    whoseTurn = COMPUTER;
} else if (board[x][y] != '*' && n < 9 && n >= 0) {
    printf("\nPosition is occupied, select any one place from the available places\n");
} else if (n < 0 || n > 8) {
    printf("Invalid position\n");
}
}

}

if (gameOver(board) == false && moveIndex == SIDE * SIDE) {
    printf("It's a draw\n");
} else {
    if (whoseTurn == COMPUTER) {
        whoseTurn = HUMAN;
    } else if (whoseTurn == HUMAN) {
        whoseTurn = COMPUTER;
    }

    declareWinner(whoseTurn);
}

```

```
}
```

```
int main() {  
    printf("\n-----");  
    printf("\t\t\t Tic-Tac-Toe\t\t\t");  
    printf("-----");  
    char cont = 'y';  
    do {  
        char choice;  
        printf("\n\nDo you want to start first? (y/n): ");  
        cin >> choice;  
        if (choice == 'n') {  
            playTTT(COMPUTER);  
        } else if (choice == 'y') {  
            playTTT(HUMAN);  
        } else {  
            printf("Invalid choice\n");  
        }  
        printf("\n\nDo you want to quit? (y/n): ");  
        cin >> cont;  
    } while (cont == 'n');  
    printf("\n\nThank you for playing <3");  
    return 0;  
}
```

