Tic Tac Toe AI Approach

Name: Bhavin Patil
Roll No.: 66

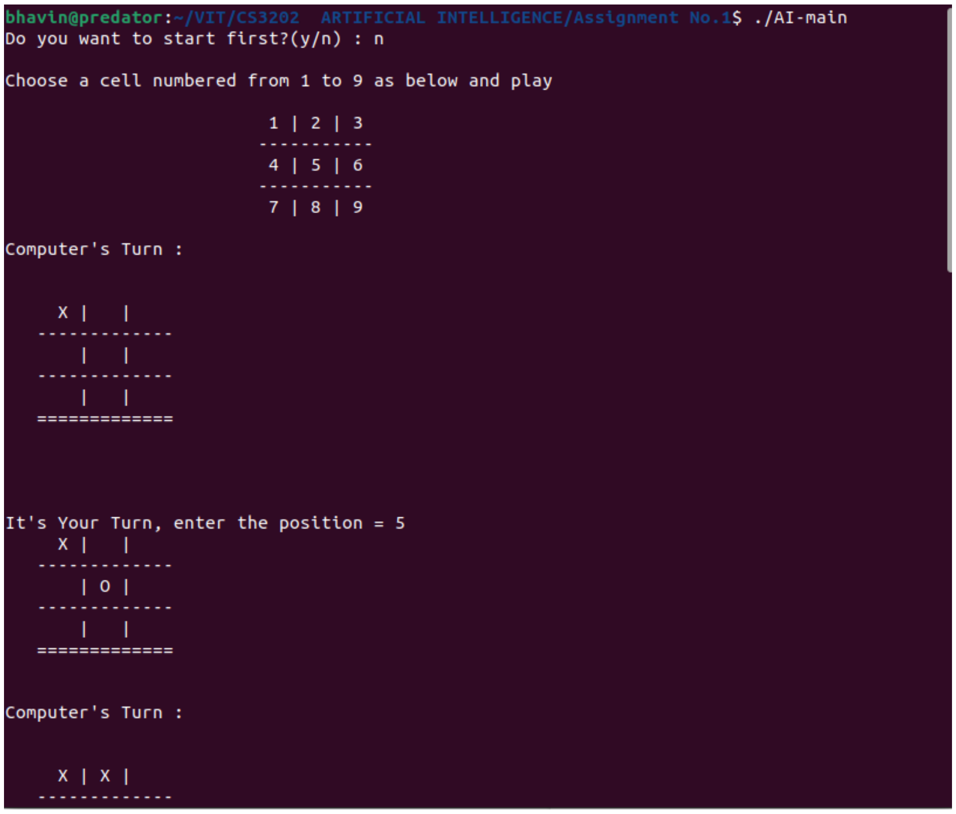
Code:

```cpp
#include <bits/stdc++.h>
using namespace std;
#define COMPUTER 1
#define USER 2
#define SIDE 3
#define COMPUTERMOVE 'X'
#define USERMOVE 'O'

// function to display the current board
void showBoard(char board[][SIDE])
{

   cout << "     " << board[0][0] << " | " << board[0][1] << " | " <<
board[0][2] << endl;
   cout << "   ------------- \n";
   cout << "     " << board[1][0] << " | " << board[1][1] << " | " <<
board[1][2] << endl;
   cout << "   ------------- \n";
   cout << "     " << board[2][0] << " | " << board[2][1] << " | " <<
board[2][2] << endl;
   cout << "   =============\n\n\n";
}

// function to display the cell index
void showInstructions()
{
   cout << "\nChoose a cell numbered from 1 to 9 as below and play\n\n";

   cout << "\t\t\t 1 | 2 | 3 \n";
   cout << "\t\t\t-----------\n";
   cout << "\t\t\t 4 | 5 | 6 \n";
   cout << "\t\t\t-----------\n";
   cout << "\t\t\t 7 | 8 | 9 \n\n";
```

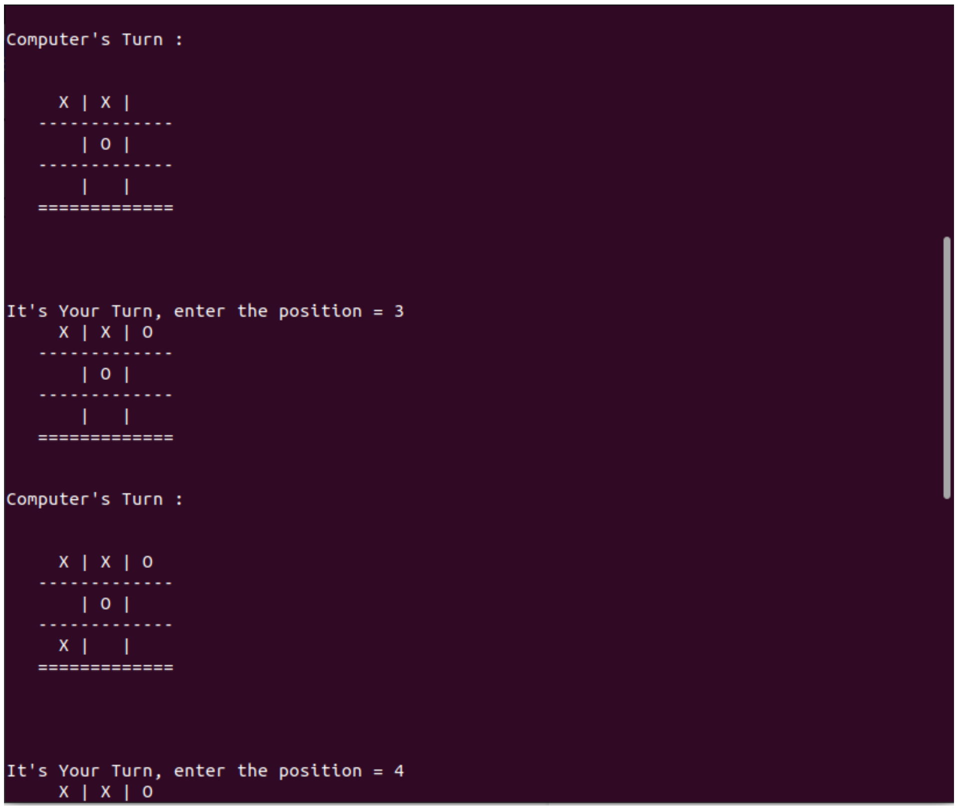
```cpp
}

// function to fill the cell with empty spaces
void initialise(char board[][SIDE])
{
   for (int i = 0; i < SIDE; i++)
   {
       for (int j = 0; j < SIDE; j++)
           board[i][j] = ' ';
   }
}

// function to declare the winner of the game
void winner(int currentPlayer)
{
   if (currentPlayer == COMPUTER)
       cout << "Loser, Computer has won!\n";
   else
       cout << "Congrates Buddy!!!, You won!\n";
}

// function to check the game is over or not also the winner of the game
the game is over
bool gameOver(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);

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

// Minimax Function to calculate best score
int minimax(char board[][SIDE], int depth, bool isAI)
{
    int score = 0;
    int bestScore = 0;
    if (gameOver(board) == true)
    {
        if (isAI == true)
            return -1;
        if (isAI == false)
            return +1;
    }
    else
    {
        if (depth < 9)
        {
            if (isAI == true)
            {
                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] = USERMOVE;
                            score = minimax(board, depth + 1, true);
                            board[i][j] = ' ';
                            if (score < bestScore)
                            {
                                bestScore = score;
                            }
                        }
                    }
                }
                return bestScore;
            }
        }
        else
        {
            return 0;
        }
    }
}

// Function to calculate best move
int bestMove(char board[][SIDE], int moves)
{
    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, moves + 1, false);
                board[i][j] = ' ';
                if (score > bestScore)
                {
                    bestScore = score;
                    x = i;
                    y = j;
                }
            }
        }
    }
    return x * 3 + y;
}

// A function to play Tic-Tac-Toe
void play(int currentPlayer)
{
    char board[SIDE][SIDE];
    int moves = 0, x = 0, y = 0;

    initialise(board);
    showInstructions();

    // Keep playing till the game is over or it is a draw
    while (gameOver(board) == false && moves != SIDE * SIDE)
    {
        int n;
        if (currentPlayer == COMPUTER)
        {
            n = bestMove(board, moves);
            x = n / SIDE;
            y = n % SIDE;
            board[x][y] = COMPUTERMOVE;
            cout << "Computer's Turn :\n\n\n";
            showBoard(board);
            moves++;
            currentPlayer = USER;
        }

        else if (currentPlayer == USER)
        {
            cout << "\n\nIt's Your Turn, enter the position = ";
            cin >> n;
            n--;
            x = n / SIDE;
            y = n % SIDE;
            if (board[x][y] == ' ' && n < 9 && n >= 0)
            {
                board[x][y] = USERMOVE;
                showBoard(board);
                moves++;
                currentPlayer = COMPUTER;
            }
            else if (board[x][y] != ' ' && n < 9 && n >= 0)
            {
                cout << "\nSike, That's position is occupied.\n\n";
            }
            else if (n < 0 || n > 8)
            {
                cout << "That's a Invalid position\n";
            }
        }
    }
    if (gameOver(board) == false && moves == SIDE * SIDE) // checking draw
condition
        cout << "That's a Drawwww!!!\n";
    else
    {
        if (currentPlayer == COMPUTER)
            currentPlayer = USER;
        else if (currentPlayer == USER)
            currentPlayer = COMPUTER;
        winner(currentPlayer);
    }
}

int main()
{
    char choice;
    cout << "Do you want to start first?(y/n) : ";
    cin >> choice;

    if (choice == 'n')
        play(COMPUTER);
    else if (choice == 'y')
        play(USER);
    else
        cout << "Invalid choice\n";

    return (0);
}
```

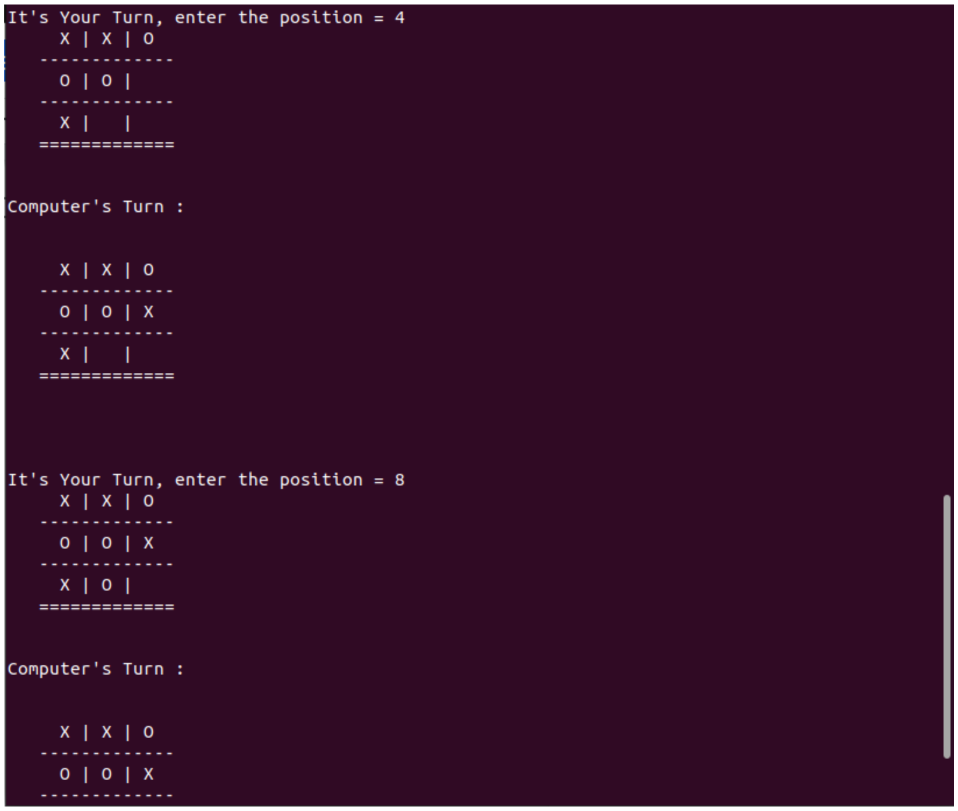
Output:



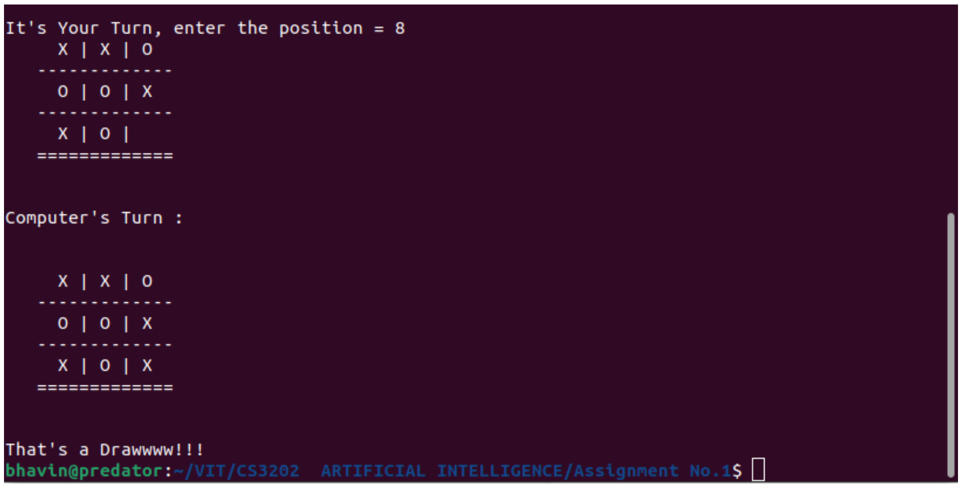
```
bhavin@predator:~/VIT/CS3202  ARTIFICIAL INTELLIGENCE/Assignment No.1$ ./AI-main
Do you want to start first?(y/n) : n

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

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

Computer's Turn :


     X |   |
   -------------
       |   |
   -------------
       |   |
   =============




It's Your Turn, enter the position = 5
     X |   |
   -------------
       | O |
   -------------
       |   |
   =============


Computer's Turn :


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



```
Computer's Turn :


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




It's Your Turn, enter the position = 3
     X | X | O
   -------------
       | O |
   -------------
       |   |
   =============


Computer's Turn :


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




It's Your Turn, enter the position = 4
     X | X | O
```



```
It's Your Turn, enter the position = 4
     X | X | O
   -------------
     O | O |
   -------------
     X |   |
   =============


Computer's Turn :


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




It's Your Turn, enter the position = 8
     X | X | O
   -------------
     O | O | X
   -------------
     X | O |
   =============


Computer's Turn :


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



```
It's Your Turn, enter the position = 8
     X | X | O
   -------------
     O | O | X
   -------------
     X | O |
   =============


Computer's Turn :


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


That's a Drawwww!!!
bhavin@predator:~/VIT/CS3202  ARTIFICIAL INTELLIGENCE/Assignment No.1$ 
```