|  |
| --- |
| **PRACTICAL FILE**  **BE (CSE) 4th Semester** |
| **Artificial Intelligence** |
| **Submitted By**  **Raghav Goyal**  **Roll Number: UE203088** |
| **Submitted To**  **Prof. Naveen Aggarwal** |
| UIET Logo  **Computer Science and Engineering**  **University Institute of Engineering and Technology**  **Panjab University, Chandigarh – 160014, INDIA**  **2022** |

# **Practical 5: Tic Tac Toe**

**Task:**

Make the Basic TIC TAC TOE Algorithm as discussed in Class.

Find Out all the ways in which algorithm can be defeated.

Make Changes in the algorithm so that it never loses the game.

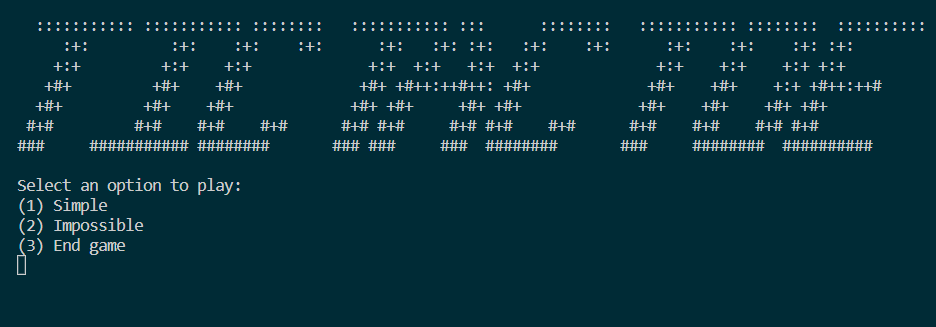
**Code:**

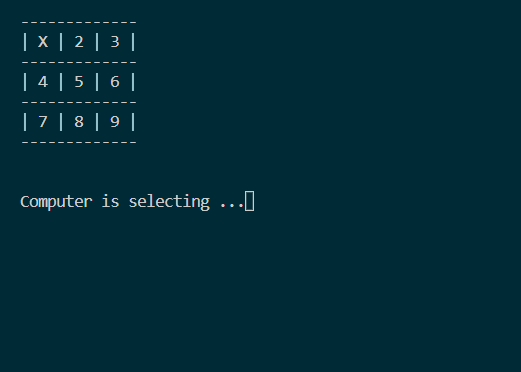
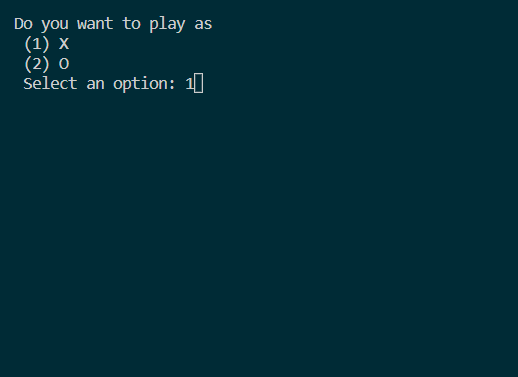
|  |
| --- |
| **#include<bits/stdc++.h>**  **using namespace std;**  **void usleep(int miliseconds){**  **std::this\_thread::sleep\_for(std::chrono::milliseconds(miliseconds));**  **}**  **int findBlank(vector<int> &board){**  **for(int i=1; i<=9; i++) if(board[i] == 2) return i;**  **}**  **void handleError(){**  **system("CLS");**  **cout << "Invalid Selection!";**  **usleep(500);**  **system("CLS");**  **}**  **int Poswin(vector<int> &board, char sign, bool winCheck){**  **int value;**  **if(!winCheck)**  **value = sign == 'X' ? 18 : 50;**  **else**  **value = sign == 'X' ? 27 : 125;**  ***//checkng rows***  **for(int i=1; i<=3; i++){**  **int mul = 1, ind = -1;**  **for(int j=1; j<=3; j++){**  **int index = (i-1)\*3 + (j);**  **mul \*= board[index];**  **if(board[index] == 2){**  **ind = index;**  **}**  **}**  **if(mul == value) return ind;**  **}**  ***//checking columns***  **for(int j=1; j<=3; j++){**  **int mul = 1, ind = -1;**  **for(int i=1; i<=3; i++){**  **int index = (i-1)\*3 + (j);**  **mul \*= board[index];**  **if(board[index] == 2){**  **ind = index;**  **}**  **}**  **if(mul == value) return ind;**  **}**  ***//checking diagonals***  **for(int k = -1; k<2; k += 2){**  **int mul = 1, ind = -1;**  **for(int i=1; i<=3; i++){**  **int j = k == -1 ? (3-i+1) : i;**  **int index = (i-1)\*3 + (j);**  **mul \*= board[index];**  **if(board[index] == 2){**  **ind = index;**  **}**  **}**  **if(mul == value) return ind;**  **}**  **return 0;**  **}**  **int make2(vector<int> &board, int turn, int corner, int difficulty){**  **int index = -1;**  **if(difficulty == 1) corner = 2;**  **if(board[5] == 2){**  **return 5;**  **}**  **else{**  **char value = turn % 2 != 0 ? 'X' : 'O';**  **for(int i=corner; i<=8; i+=2){**  **if(board[i] == 2){**  **board[i] = turn % 2 != 0 ? 3 : 5;**  **if(Poswin(board, value, 0)){**  **board[i] = 2;**  **return i;**  **}**  **}**  **}**  **}**  **return index;**  **}**  **void Go(vector<int> &board, int n, int &turn){**  **board[n] = turn%2 != 0 ? 3 : 5;**  **turn++;**  **}**  **void displayBoard(vector<int> &board){**  **for(int i=0; i<13; i++) cout << '-'; cout << endl;**  **for(int i=1; i<=3; i++){**  **cout << '|';**  **for(int j=1; j<=3; j++){**  **int index = (i-1)\*3 + (j);**  **char sign = char(index + int('0'));**  **if(board[index] == 3){**  **sign = 'X';**  **}**  **else if(board[index] == 5){**  **sign = 'O';**  **}**  **cout << ' ' << sign << " |";**  **}**  **cout << endl;**  **for(int i=0; i<13; i++) cout << '-'; cout << endl;**  **}**  **}**  **string Title(){**  **string line1 = " ::::::::::: ::::::::::: :::::::: ::::::::::: ::: :::::::: ::::::::::: :::::::: ::::::::::\n";**  **string line2 = " :+: :+: :+: :+: :+: :+: :+: :+: :+: :+: :+: :+: :+:\n";**  **string line3 = " +:+ +:+ +:+ +:+ +:+ +:+ +:+ +:+ +:+ +:+ +:+\n";**  **string line4 = " +#+ +#+ +#+ +#+ +#++:++#++: +#+ +#+ +#+ +:+ +#++:++#\n";**  **string line5 = " +#+ +#+ +#+ +#+ +#+ +#+ +#+ +#+ +#+ +#+ +#+\n";**  **string line6 = " #+# #+# #+# #+# #+# #+# #+# #+# #+# #+# #+# #+# #+#\n";**  **string line7 = "### ########### ######## ### ### ### ######## ### ######## ##########\n";**  **return line1+line2+line3+line4+line5+line6+line7;**  **}**  **bool playTurn(vector<int> &board, int &turn, int &diffculty){**  **cout << endl << endl;**  **cout << "Computer is selecting ";**  **for(int i=0; i<3; i++) cout << '.', usleep(300);**  **for(int i=0; i<2; i++) cout << "\b \b", usleep(300);**  **switch(turn){**  **case 1:{**  **Go(board, 1, turn);**  **break;**  **}**  **case 2:{**  **board[5] == 2 ? Go(board, 5, turn) : Go(board, 1, turn);**  **break;**  **}**  **case 3:{**  **board[9] == 2 ? Go(board, 9, turn) : Go(board, 3, turn);**  **break;**  **}**  **case 4:{**  **int winX = Poswin(board, 'X', 0);**  **int corner = 2;**  **if(board[5] == 3) corner = 1;**  **winX != 0 ? Go(board, winX, turn) : Go(board, make2(board, turn, corner, diffculty), turn);**  **break;**  **}**  **case 5:{**  **int winX = Poswin(board, 'X', 0), winO = Poswin(board, 'O', 0);**  **if(winX != 0){**  **Go(board, winX, turn);**  **return true;**  **}**  **else if(winO != 0){**  **Go(board, winO, turn);**  **}**  **else{**  **board[7] == 2 ? Go(board, 7, turn) : Go(board, 3, turn);**  **}**  **break;**  **}**  **case 6:{**  **int winX = Poswin(board, 'X', 0), winO = Poswin(board, 'O', 0);**  **if(winO != 0){**  **Go(board, winO, turn);**  **return true;**  **}**  **else if(winX != 0){**  **Go(board, winX, turn);**  **}**  **else{**  **int corner = 2;**  **if(board[5] == 3) corner = 1;**  **Go(board, make2(board, turn, corner, diffculty), turn);**  **}**  **break;**  **}**  **case 8:{**  **int winX = Poswin(board, 'X', 0), winO = Poswin(board, 'O', 0);**  **if(winO != 0){**  **Go(board, winO, turn);**  **return true;**  **}**  **else if(winX != 0){**  **Go(board, winX, turn);**  **}**  **else{**  **Go(board, findBlank(board), turn);**  **}**  **break;**  **}**  **case 7:**  **case 9:{**  **int winX = Poswin(board, 'X', 0), winO = Poswin(board, 'O', 0);**  **if(winX != 0){**  **Go(board, winX, turn);**  **return true;**  **}**  **else if(winO != 0){**  **Go(board, winO, turn);**  **}**  **else{**  **Go(board, findBlank(board), turn);**  **}**  **break;**  **}**  **}**  **return false;**  **}**  **bool playHuman(vector<int> &board, int &turn){**  **cout << endl << endl;**  **cout << "Enter index where you want to tick: ";**  **int option;**  **cin >> option;**  **if(option < 1 || option > 9 || board[option] != 2) return false;**  **char sign = turn % 2 == 0 ? 'O' : 'X';**  **Go(board, option, turn);**  **if(Poswin(board, sign, 1)) return true;**  **return false;**  **}**  **void displayWinner(int winner, vector<int> &board){**  **system("CLS");**  **displayBoard(board);**  **if(winner == 0){**  **cout << "YOU LOSE!";**  **}**  **else if(winner == 1){**  **cout << "YOU WIN!";**  **}**  **else{**  **cout << "DRAW!";**  **}**  **cout << "\n\n\n\nPress any button to continue!\n";**  **char x;**  **cin >> x;**  **}**  **void playGame(int &difficulty){**  **vector<int> board(10, 2);**  **int turn = 1;**  **int option = -1;**  **while(true){**  **cout << "Do you want to play as\n (1) X\n (2) O\n Select an option: ";**  **cin >> option;**  **if(option == 1 || option == 2){**  **option--;**  **break;**  **}**  **handleError();**  **}**  **int win = -1;**  **while(turn < 10){**  **system("CLS");**  **displayBoard(board);**  **if(((turn ^ option) & 1) == 1){**  **win = playHuman(board, turn);**  **if(win){**  **displayWinner(1, board);**  **return;**  **}**  **win = -1;**  **}**  **else{**  **win = playTurn(board, turn, difficulty);**  **if(win){**  **displayWinner(0, board);**  **return;**  **}**  **win = -1;**  **}**  **}**  **if(win == -1) displayWinner(-1, board);**  **}**  **int main(){**  **while(true){**  **system("CLS");**  ***// Game Starts***  **string name = Title();**  **cout << name << endl;**  **cout << "Select an option to play: \n(1) Simple \n(2) Impossible \n(3) End game\n";**  **int difficulty;**  **cin >> difficulty;**  **system("CLS");**  **if(difficulty == 1 || difficulty == 2){**  **playGame(difficulty);**  **}**  **else if(difficulty == 3){**  **return 0;**  **}**  **else{**  **handleError();**  **}**  **}**  **}** |

**Output:**

Video link: [Drive link to video](https://drive.google.com/file/d/1kfTE6ETemwRw03idk23XGsyqdmHk_YFy/view?usp=sharing)

Images:





**Where the algorithm fails:**

Possible cases to win when player places X are:

|  |
| --- |
| Case #1: [8, 6, 9, 3]  -------------  | 1 | O | X |  -------------  | 4 | O | X |  -------------  | O | X | X |  -------------  Case #2: [4, 2, 1, 7]  -------------  | X | X | O |  -------------  | X | O | O |  -------------  | X | 8 | 9 |  -------------  Case #3: [4, 2, 9, 1, 7]  -------------  | X | X | O |  -------------  | X | O | O |  -------------  | X | O | X |  -------------  Case #4: [8, 4, 7, 1]  -------------  | X | O | 3 |  -------------  | X | O | 6 |  -------------  | X | X | O |  -------------  Case #5: [8, 4, 3, 7, 1]  -------------  | X | O | X |  -------------  | X | O | O |  -------------  | X | X | O |  -------------  Case #6: [8, 6, 1, 9, 3]  -------------  | X | O | X |  -------------  | O | O | X |  -------------  | O | X | X |  -------------  Case #7: [6, 2, 3, 9]  -------------  | O | X | X |  -------------  | O | O | X |  -------------  | 7 | 8 | X |  -------------  Case #8: [7, 2, 6, 3, 9]  -------------  | O | X | X |  -------------  | O | O | X |  -------------  | X | O | X |  -------------  Case #9: [5, 9, 3, 7]  -------------  | O | O | X |  -------------  | 4 | X | O |  -------------  | X | 8 | X |  ------------- |

Possible cases to win when player places O are: None

**Changes in the algorithm so that it never loses the game:**

The computer would always win if it starts first. In case it starts second, the following changes were incorporated.

Changed make2() function to return corner if player1 starts with the center and return edge if player1 starts with a corner.

Also, made the make2() function such that, amongst all possible blank spaces, the algorithm checks if placing current character in it would result in Poswin() function returning a non-zero value for current character.

This lookahead functionality results in computer never losing the game whether it starts first or second.

Previous function:

|  |
| --- |
| **int** make2(vector<**int**> **&**board){  if(board[5] == 2){  return 5;  }  else{  for(**int** i=2; i<=8; i+=2){  if(board[i] == 2){  return i;  }  }  }  } |

Improved function:

|  |
| --- |
| **int** make2(vector<**int**> **&**board, **int** turn, **int** corner, **int** difficulty){  **int** index = -1;  if(difficulty == 1) corner = 2;  if(board[5] == 2){  return 5;  }  else{  **char** value = turn % 2 != 0 ? 'X' : 'O';  for(**int** i=corner; i<=8; i+=2){  if(board[i] == 2){  board[i] = turn % 2 != 0 ? 3 : 5;  if(Poswin(board, value, 0)){  board[i] = 2;  return i;  }  }  }  }  return index;  } |