README:

The following document contains our program code and a proof of our method to solve question 1 of the assignment.

The proof we have given is using FOL and is listed after this page.

The program code comes after the proof

To run the Program:
1. Copy code into file.
2.Compile with g++ and run.

Group Members:

1. Kshitij R Dani – 2015B4A70594G
2. MVS Chaitanya Mukka – 2015B4A70536G
3. Aiswarya Subramanian – 2015B4A70585G
4. Shivin Thukral - 2015B4A70350G

TIC TAC TOE

I. INTRODUCTION :

The purpose of this assignment is to change the rule of tic-tac-toe that would enable the player 1 to win always. A mathematical proof using first Order Logic (FOL) and an exhaustive proof of the same is also given. This has been implemented in C++; which is given at the end.

II. RULES OF THE GAME :

Above the existing rules of tic-tac-toe, the following rules have been added to assure a win for the first player:

→ The players cannot play at the adjacent position (ie. vertically or horizontally adjacent cells) unless:

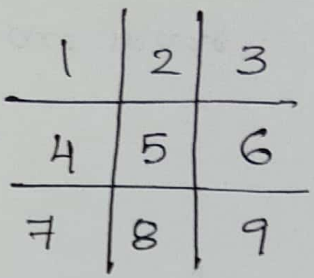
① He can form a triplet to win

② If all other cells other than the adjacent cells are already occupied.

→ Also the following assumptions are made:

① We assume that both the players play optimally that is the player plays to win or try to force a draw. (More about optimality is discussed in next section).

② From point ①; the first player plays from cell 1 (which is shown to be optimal in section III) without loss of generality.



With respect to cell 1, the cells (2,4), (3,7) and (6,8) are interchangeable (due to symmetry).

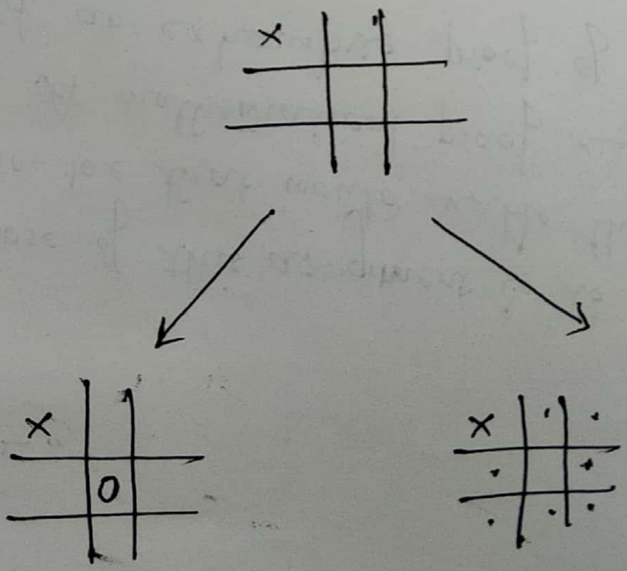
→ The following are the notations used henceforth :

① X(i) ⇒ denotes that the first player plays his move at the ith cell.

② O(i) ⇒ denotes that the second player plays his move at the ith cell.

③ W(A) ⇒ denotes that A wins the game. 'A' can be X (the first player) (or) O (the second player).

→ The following tree denotes the exhaustive proof.



Leaf 1 : If player 2 plays at position 5

Leaf 2 : If player 2 plays at positions other than 5.

Scanned by CamScanner

III. OPTIMALITY CONDITIONS :

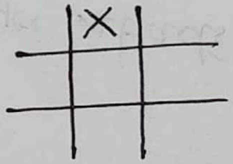
As discussed before; an optimal move refers to either a player winning the game or forcing a draw. The sequence of moves of a player is said to be sub-optimal if it leads to draw or a win.

① Corner is the optimal position :

We shall show that the positions 2, 4, 6, 8 and 5 are not optimal.

(i) The 2, 4, 6, 8 case :

By symmetry, the positions 2, 4, 6 and 8 are same for the player 1 to start. Hence we discuss only about X (2) case.



To show that this is not an optimal strategy for X; we shall show one game exists where O wins or the game leads to a draw.

Consider the following two scenarios :

⊢ X (2)	⊢ X (2)
⊢ O (5)	⊢ O (5)
⊢ X (7)	⊢ X (4)
⊢ O (9)	⊢ O (7)
⊢ X (6)	⊢ X (9)
⊢ O (1)	⊢ O (3)
W (O)	W (O)

{ Also, we get similar cases when 7 is interchanged with 9 }

In both scenarios; both players play optimally and follow the rule imposed.

(ii) The 5 case

Consider the following case :

$\vdash X(5)$
$\vdash O(1)$
$\vdash X(3)$ {X(2,4,6,8) are not possible due to the rule}
$\vdash O(7)$
$\vdash X(9)$
$\vdash O(4)$

W(0)

Hence we conclude that the first player starts from corner cells. The next section shows why the cells 1, 3, 7 and 9 are the optimal cases.

② Suboptimal cases :

We shall show that; even if first player starts at the corner cell; and if his next consecutive move is not in the corner cells; the game may lead to a draw / losing it

$\vdash X(1)$
$\vdash O(4)$
$\vdash X(6)$
$\vdash O(3)$
$\vdash X(7)$
$\vdash O(9)$.

$\vdash X(8)$
$\vdash O(2)$
$\vdash X(5)$

DRAW

Hence the first player doesnot choose these strategies to play.

IV. THE STRATEGY :

The following cases show the optimal strategies by which player 1 wins ; depending on the second player's first move.

Case 1 : ⊢ X(1) { first move }

The first move of 'O' is 2/4 { interchangeable positions }

STRATEGY :

```
⊢ X(1)                ⊢ X(1)
⊢ O(2)                ⊢ O(4)
⊢ X(7)       (OR)     ⊢ X(3)
⊢ O(4)                ⊢ O(2)
⊢ X(9)                ⊢ X(9)
⊢ O(8)                ⊢ O(6)
⊢ X(5)                ⊢ X(5)
--------              --------
 W(X)                  W(X)
```

Case 2 : The first move of O is 7/3

STRATEGY :

```
⊢ X(1)
⊢ O(7)
⊢ X(3)
⊢ O(2)
⊢ X(9)
⊢ O(6)
⊢ X(5)
--------
 W(X)
```

Case 3 : The first move of O is 6/8

STRATEGY : $\vdash X(1)$
$\vdash O(6)$
$\vdash X(3)$
$\vdash O(2)$
$\vdash X(7)$
$\vdash O(4)$
$\vdash X(5)$

$W(X)$

Case 4 : The first move of O is 9

STRATEGY : $\vdash X(1)$
$\vdash O(9)$
$\vdash X(3)$
$\vdash O(2)$
$\vdash X(7)$
$\vdash O(4)$
$\vdash X(5)$

$W(X)$

Case 5 : The first move of O is 5

STRATEGY : $\vdash X(1)$
$\vdash O(5)$
$\vdash X(3)$
$\vdash O(7)$
$\vdash X(2)$

$W(X)$

THE PROGRAM

```
#include<iostream>


char board[]={' ',' ',' ',' ',' ',' ',' ',' ',' '};

int O1[]={0,1,7,9,3,0,3,9,7,3}; // Positions Player1 should play after Player2's first move

int O2[][2]={0,0,0,0,9,7,0,0,9,3,7,3};// Positions Player1 should play after Player2's second
move

int PosCheck(int pos){


    if(board[pos-2]=='O'||board[pos]=='O'||board[pos+2]=='O'||board[pos-4]=='O'){
        if(board[pos-2]!=' '&&board[pos-4]!=' '&&board[pos+2]!=' '&&board[pos+4]!=' ')
               return 0;
        else                  return 1;
        }
    else
        return 0;
}



int check(int i, int j){
    int sum=0,pos,space=0;
    for(int k=0;k<3;k++){
        if(board[i+j*k-1]=='X')
               sum++;
        if(board[i+j*k-1]==' '){
               pos=i+j*k;
               space=1;
        }

    }

    if(sum==2 && space==1)
        return pos;
    else
        return 0;
}

int winCheck(){

    int i;
    for(i=1;i<8;i=i+3)
        if(check(i,1)!=0){

                return check(i,1);
        }

    for(i=1;i<4;i++)
        if(check(i,3)!=0){

                return check(i,3);
        }

    if(check(1,4)!=0){

        return check(1,4);
    }
    else if(check(3,2)!=0){

        return check(3,2);
    }
    else
        return 0;


}




void displayBoard(){
    std::cout<<"
"<<board[0]<<"|"<<board[1]<<"|"<<board[2]<<"\n";
    std::cout<<"                                   "<<"-------"<<"\n";
    std::cout<<"
"<<board[3]<<"|"<<board[4]<<"|"<<board[5]<<"\n";
    std::cout<<"                                   "<<"-------"<<"\n";
    std::cout<<"
"<<board[6]<<"|"<<board[7]<<"|"<<board[8]<<"\n";
}

int main(){

    std::cout<<"Hello there. Computer Player1 and Mark is 'X'. You are Player2 and Mark is
'O'.\n\n";

    displayBoard();
    std::cout<<"\n\n";
    bool play=true;
```

```
int pos2=1,moves=0,pos1;

board[0]='X';
do{

    std::cout<<"\n\nPlayer1 has put his mark in position "<<O1[pos2]<<"\n\n";
    moves++;

    displayBoard();

    L1:
    std::cout<<"\nEnter position number where you wish to place mark(1-9)\n";
    std::cin>>pos2;


    if(moves==3||moves==5){
           if(PosCheck(pos2)==1){
                  std::cout<<"\nFollow rules!!!!";
                  goto L1;
           }
    }
    // IF PLAYER2'S FIRST MOVE IS IN THE CENTER THEN GOTO L1
    if(moves==1 && pos2==5){

           board[pos2-1]='O';
           goto L2;
    }

    if(board[pos2-1]=='X' || board[pos2-1]=='O'){
           std::cout<<"\nThis cell is already occupied. Place somewhere else\n";
           goto L1;
    }
    else{
           board[pos2-1]='O';
           moves++;
    }


    displayBoard();


    //CHECKING TO SEE IF A WINNING COMBINATION HAS BEEN ESTABLISHED
    if(winCheck()!=0){
           std::cout<<"\nPlayer1 wins by putting X at position "<<winCheck()<<"\n";
           board[winCheck()-1]='X'; //Putting X to show player2 the winning combination
           /**/displayBoard();
           play=false;
    }
    //PLAYER1'S SECOND MOVE CORRESPONDING TO PLAYER2'S FIRST MOVE
    if(moves==2){
           board[O1[pos2]-1]='X';
    }


    //PLAYER1'S THIRD MOVE CORRESPONDING TO PLAYER2'S SECOND MOVE
    if(moves==4){
           if((board[1]=='O'&&board[3]=='O')||(board[2]=='O'&&board[4]=='O')){

                  board[O2[pos2][0]-1]='X';

           }
           else{
                  board[O2[pos2][1]-1]='X';

           }
    }







}while(play==true);
goto L5;

//CASE WHEN PLAYER2'S FIRST MOVE IS IN THE MIDDLE

L2:


board[2]='X';
std::cout<<"\n\nPlayer1 has put his mark in position 3"<<"\n\n";
moves++;

/**/displayBoard();

L3:
std::cout<<"\nEnter position number where you wish to place mark3(1-9)\n";
std::cin>>pos2;

if(PosCheck(pos2)==1){
                  std::cout<<"\nFollow rules!!!!";
                  goto L3;
}

std::cout<<"\nPlayer1 wins by putting X at position "<<winCheck()<<"\n";
board[winCheck()-1]='X'; //Putting X to show player2 the winning combination
```

```
    displayBoard();



    L5:

    return 0;
}
```