PAGE NO :
DATE :

Tic Tac Toe

- ## Algorithm

1. **Initialize Board**

   Create a dictionary board with keys from 1 to 9. Set all values of board to ' '

2. **Print the Board**

Function printBoard (board):

print the current state of the board in a 3x3 grid format

3. **player's turn**

Function playerMove (board)

Do:

prompt the player to input a move (1-9)

While the choosen position not ~~free space~~ (SpaceFree (board, move) == False)

Insert 'O' at the choosen position (insertLetter (board, 'O', move)

If checkWin (board, 'O'):

Print "You Win"!

End game

If checkDraw (board):

Print "It's a draw!"

End game

4. Bot's Turn (Minmax Algorithm)

Function botMove (board):
    Call minmax (board, isMaximizing = True)
    Get the best possible move for 'x' from the Minmax algorithm.
    Insert 'x' at the position (insertLetter (board, 'x', bestMove))
    If checkWin (board, 'x'):
        Print "Bot wins!"
        End game
    If checkDraw (board):
        Print "Its a draw!"
        End game

5. MinMax Algorithm

Function minmax (board, isMaximizing)
    If checkWin (board, 'x'):
        Return score = 1
    If checkWin (board, 'o'):
        Return score = -1
    If checkDraw (board)
        Return score = 0.

If isMaximizing == True
    Initialize bestScore = -∞
    For each empty position on the board:
        Simulate placing 'x' on that position
        Call minmax (board, isMaximizing = True) recursively

        Undo the move

Update bestScore to the minimum of current bestScore and the returned score.
Return bestScore.

6. Insert a letter

Function insertLetter(board, letter, position):
place the given ('X' or 'O') at the specified position on the board.

7. Check if space is free

Function SpaceFree(board, position)
Return true if the board position is empty, else return False

8. Check for win

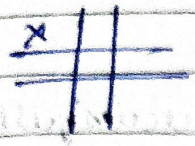
9. Check for a Draw

10. Game Loop.

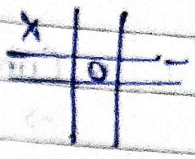
Function main():
Initialize board as using InitializeBoard()
While game is not over:
call printBoard(board)
Call playerMove(board)
call botMove(board)
End game when a win or draw is detected

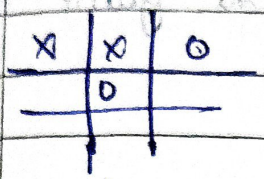
PAGE NO :
DATE :

OUTPUT

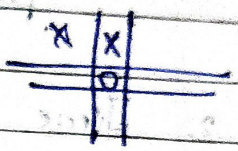


Enter position for O: 5

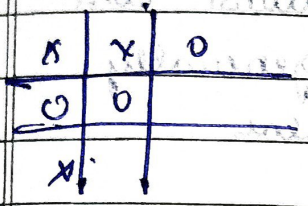


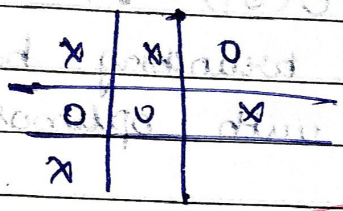


Enter position for O: 2

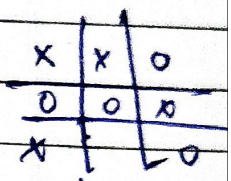


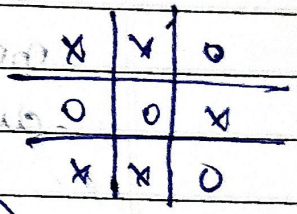
Enter position for O: 4





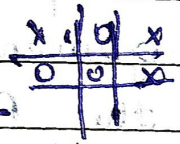
Enter for position O: 3

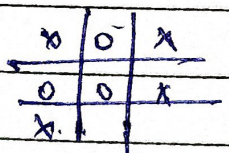


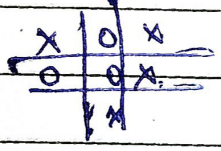


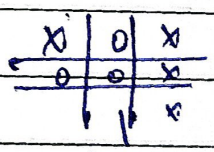
Draw!

State Space tree

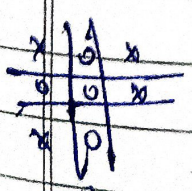




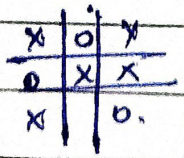


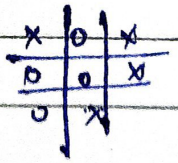


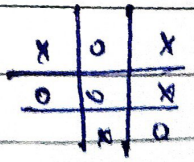
+1.

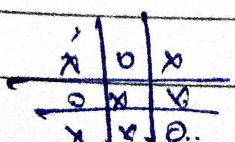


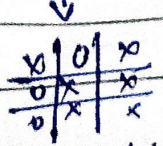
-1



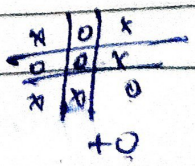








+1.



+0

PAGE NO :
DATE :

Evaluation Steps

1. Completeness
   -> The Implementation successfully checks for wins, draws, and manages player turns until the game concludes

2. Time Complexity
   MinMax Algorithm
   O(9!) in the worst case due to branching but generally faster with optimal play

3. Space complexity
   O(1) for the board, with a maximum call stack depth of O(9) during minmax recursion.

4. Optimal.
   The minmax algorithm ensures optimal play, guarantees the best outcome for the AI