**Artificial Intelligence Lab**

**Name** - N Aditya Sai

**Register Number**- RA1911030010075

**Topic**: Constraint Satisfaction Problem

**Experiment** – 6

**Aim:** To study and implement applications of Minimax Algorithm

**Application:** Tic Tac Toe

**Code:**

theBoard = {'1': ' ' , '2': ' ' , '3': ' ' ,

'4': ' ' , '5': ' ' , '6': ' ' ,

'7': ' ' , '8': ' ' , '9': ' ' }

board\_keys = []

for key in theBoard:

board\_keys.append(key)

def printBoard(board):

print(board['7'] + '|' + board['8'] + '|' + board['9'])

print('-+-+-')

print(board['4'] + '|' + board['5'] + '|' + board['6'])

print('-+-+-')

print(board['1'] + '|' + board['2'] + '|' + board['3'])

def game():

turn = 'X'

count = 0

for i in range(10):

printBoard(theBoard)

print("It's your turn," + turn + ".Move to which place?")

move = input()

if theBoard[move] == ' ':

theBoard[move] = turn

count += 1

else:

print("That place is already filled.\nMove to which place?")

continue

if count >= 5:

if theBoard['7'] == theBoard['8'] == theBoard['9'] != ' ':

printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\*\*\* " +turn + " won. \*\*\*\*")

break

elif theBoard['4'] == theBoard['5'] == theBoard['6'] != ' ':

printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\*\*\* " +turn + " won. \*\*\*\*")

break

elif theBoard['1'] == theBoard['2'] == theBoard['3'] != ' ': # across the bottom

printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\*\*\* " +turn + " won. \*\*\*\*")

break

elif theBoard['1'] == theBoard['4'] == theBoard['7'] != ' ': # down the left side

printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\*\*\* " +turn + " won. \*\*\*\*")

break

elif theBoard['2'] == theBoard['5'] == theBoard['8'] != ' ': # down the middle

printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\*\*\* " +turn + " won. \*\*\*\*")

break

elif theBoard['3'] == theBoard['6'] == theBoard['9'] != ' ': # down the right side

printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\*\*\* " +turn + " won. \*\*\*\*")

break

elif theBoard['7'] == theBoard['5'] == theBoard['3'] != ' ': # diagonal

printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\*\*\* " +turn + " won. \*\*\*\*")

break

elif theBoard['1'] == theBoard['5'] == theBoard['9'] != ' ': # diagonal

printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\*\*\* " +turn + " won. \*\*\*\*")

break

# If neither X nor O wins and the board is full, we'll declare the result as 'tie'.

if count == 9:

print("\nGame Over.\n")

print("It's a Tie!!")

# Now we have to change the player after every move.

if turn =='X':

turn = 'O'

else:

turn = 'X'

# Now we will ask if player wants to restart the game or not.

restart = input("Do want to play Again?(y/n)")

if restart == "y" or restart == "Y":

for key in board\_keys:

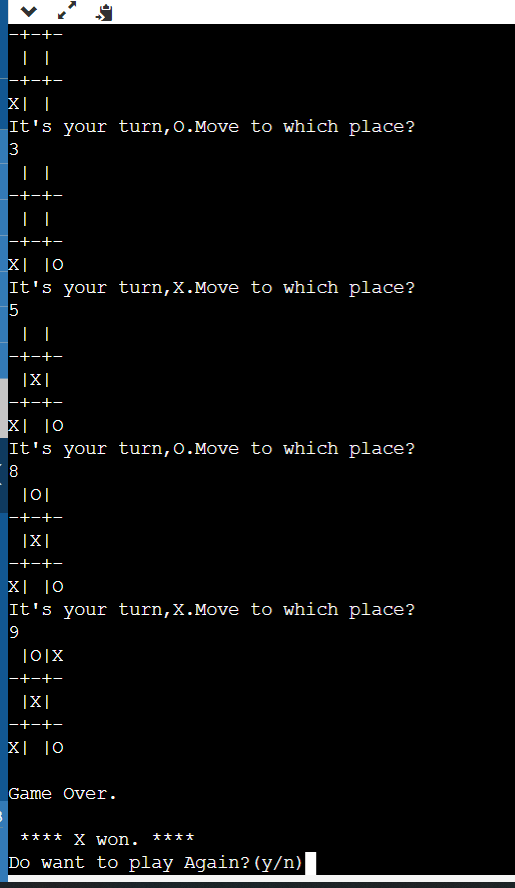
theBoard[key] = " "

game()

if \_\_name\_\_ == "\_\_main\_\_":

game()

**Output:**



**Result:** Minimax algorithm was studied and applied in a Tic Tac Toe game