**Artificial Intelligence**

**A logo of a university

Description automatically generated**

**CEP Report**

By

|  |  |  |
| --- | --- | --- |
| Name | Reg.# | Section |
| Abdullah Sajawal | CIIT/FA22-BCE-089/ATD | BCE-5B |

**Date:** 27 Dec 2024

**Submitted To:**

Dr. Shoaib Azmat

Department of Electrical & Computer Engineering

(Fall 2024 Session)

COMSATS University Islamabad, Abbottabad Campus

**Tic Tac Toe:**

**Task:**

Implement Tic Tac Toe game in Python using mini max algorithm.

**Code:**

board = {1:" ",2:" ",3:" ",4:" ",5:" ",6:" ",7:" ",8:" ",9:" "}

def print\_board():

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

print("--+---+--")

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

print("--+---+--")

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

print()

def check\_for\_win(player):

return ((board[1] == board[2] == board[3] == player) or

(board[4] == board[5] == board[6] == player) or

(board[7] == board[8] == board[9] == player) or

(board[1] == board[4] == board[7] == player) or

(board[2] == board[5] == board[8] == player) or

(board[3] == board[6] == board[9] == player) or

(board[1] == board[5] == board[9] == player) or

(board[3] == board[5] == board[7] == player))

def check\_for\_draw():

return all(space != " " for space in board.values())

def minimax(board, is\_maximizing):

if check\_for\_win("o"):

return 1

if check\_for\_win("x"):

return -1

if check\_for\_draw():

return 0

if is\_maximizing:

best\_score = -float('inf')

for key in board.keys():

if board[key] == " ":

board[key] = "o"

score = minimax(board, False)

board[key] = " "

best\_score = max(best\_score, score)

return best\_score

else:

best\_score = float('inf')

for key in board.keys():

if board[key] == " ":

board[key] = "x"

score = minimax(board, True)

board[key] = " "

best\_score = min(best\_score, score)

return best\_score

def play\_computer():

best\_score = -float('inf')

best\_move = None

for key in board.keys(): #checking all empty positions

if board[key] == " ":

board[key] = "o"

score = minimax(board, False)

board[key] = " " #Backtracking

if score > best\_score:

best\_score = score

best\_move = key

board[best\_move] = "o"

def play\_game():

turn = "x"

while True:

print\_board()

if check\_for\_win("x"):

print("Player X wins!")

break

if check\_for\_win("o"):

print("AI wins!")

break

if check\_for\_draw():

print("It's a draw!")

break

if turn == "x":

move = int(input("Player X, enter your move (1-9): "))

if board[move] == " ":

board[move] = "x"

turn = "o"

else:

print("Invalid move. Try again.")

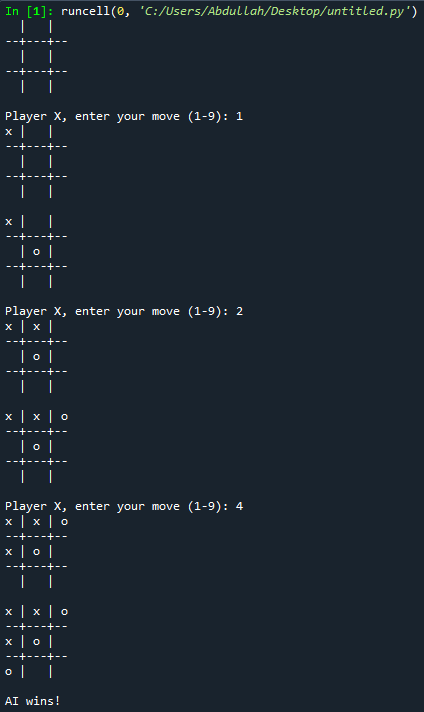
else:

play\_computer()

turn = "x"

play\_game()

**Output:**

****