**CSE422\_06\_Lab\_Assignment03\_Turnitin\_Summer2024**

**Name :** TAHMID  **| Student ID :** 21201701

print("name: TAHMID IQBAL")

print("ID: 21201701")

Part 01

**import** random

def *round\_wins*():

**return** random.choice([1, -1])

def *minimax*(*dep*, *figher*, α, β, *maxi*):

**if** dep == 5:

**return** round\_wins()

**if** maxi:

max\_eval = -float('inf')

**for** \_ in range(2):

eval = minimax(dep + 1, figher, α, β, False)

max\_eval = max(max\_eval, eval)

α = max(α, eval)

**if** β <= α:

**break**

**return** max\_eval

**else**:

min\_eval = float('inf')

**for** \_ in range(2):

eval = minimax(dep + 1, figher, α, β, True)

min\_eval = min(min\_eval, eval)

β = min(β, eval)

**if** β <= α:

**break**

**return** min\_eval

def *play\_game*(*starting\_figher*):

rounds = 5

round\_winners = []

current\_figher = starting\_figher

scorpion\_wins = 0

subzero\_wins = 0

**for** round\_num in range(1, rounds + 1):

maxi = (current\_figher == 1)

round\_winner = minimax(0, current\_figher, -float('inf'), float('inf'), maxi)

**if** round\_winner == 1:

round\_winners.append("Sub-Zero")

subzero\_wins += 1

**else**:

round\_winners.append("Scorpion")

scorpion\_wins += 1

# If any figher wins 3 rounds ; we break no need for further matches

**if** scorpion\_wins == 3 or subzero\_wins == 3:

**break**

current\_figher = 1 - current\_figher

**if** scorpion\_wins > subzero\_wins:

game\_winner = "Scorpion"

**else**:

game\_winner = "Sub-Zero"

**return** game\_winner, round\_winners

starting\_figher = int(input("Enter who starts the battle (0 for Scorpion, 1 for Sub-Zero): "))

game\_winner, round\_winners = play\_game(starting\_figher)

print(f"Game Winner: *{game\_winner}*")

print(f"Total Rounds Played: {len(round\_winners)}")

**for** i, winner in enumerate(round\_winners, 1):

print(f"Round *{i}* wins: *{winner}*")

PART 02

def *minimax*(*node*, *depth*, *alpha*, *beta*, *mini\_player*, *outcome\_ls*):

**if** depth == 0:

**return** outcome\_ls[node]

**if** mini\_player:

max\_eval = float('-inf')

**for** i in range(2):

eval = minimax(node \* 2 + i, depth - 1, alpha, beta, False, outcome\_ls)

max\_eval = max(max\_eval, eval)

alpha = max(alpha, eval)

**if** beta <= alpha:

**break**

**return** max\_eval

**else**:

min\_eval = float('inf')

**for** i in range(2):

eval = minimax(node \* 2 + i, depth - 1, alpha, beta, True, outcome\_ls)

min\_eval = min(min\_eval, eval)

beta = min(beta, eval)

**if** beta <= alpha:

**break**

**return** min\_eval

## here the minimax func calculates the minimax values of the game recusively

## as well as implements the alpha, beta pruning

def *mypacman\_game*(*c*):

outcome\_ls = [3, 6, 2, 3, 7, 1, 2, 0]

minimax\_val = minimax(0, 3, float('-inf'), float('inf'), True, outcome\_ls)

dark\_magic\_left = max(outcome\_ls[0:4]) - c

dark\_magic\_right = max(outcome\_ls[4:8]) - c

zero\_dark\_magic = minimax\_val

using\_dark\_magic = max(dark\_magic\_left, dark\_magic\_right)

**if** using\_dark\_magic > zero\_dark\_magic:

**return** f"The new minimax value is *{using\_dark\_magic}*. Pacman uses dark magic."

**else**:

**return** f"The minimax value is *{zero\_dark\_magic}*. Pacman does not use dark magic."

print(mypacman\_game(1))

print(mypacman\_game(2))

print(mypacman\_game(5))

PART 03

# Is the first player always a maximizer node?

#ans: it all depends on the specific game context and the roles assigned to the players at the init.

# eg. If the first player is the one trying to maximize their utility, they are the maximizer.

# Can alpha-beta pruning handle stochastic environments?

#ans: No, alpha-beta pruning is not designed for stochastic environments. Stochastic environments deals with rolling dice, drawing cards, or random events etc. However, alpha-beta pruning relies on the assumption that there are only two types of nodes (maximizing and minimizing)