**Task no 01**

f1 = open("20101539\_Umme Abira Azmary\_CSE422\_11\_Assignment03\_Summer2024\_InputFile1.txt", "r")

import random

def AB\_pruning(depth, state, alpha, beta):

child = 2

given\_depth = 5

if depth == given\_depth:

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

if state:

eval = -10000000000000 #negative infinity

for x in range(child):

val = AB\_pruning(depth + 1, False, alpha, beta)

eval = max(eval, val)

alpha = max(alpha, eval)

if beta <= alpha:

break

return eval

else:

eval = +100000000000000 #positive infinity

for x in range(child):

val = AB\_pruning(depth + 1, True, alpha, beta)

eval = min(eval, val)

beta = min(beta, eval)

if beta <= alpha:

break

return eval

#print(AB\_pruning(0, True, -100000, 100000000))

def game\_playing(player1):

results = []

scorpio = 0

subzero = 0

rounds = 3

c\_player = None

if player1 == 0:

c\_player = False

else:

c\_player = True

for x in range(1, rounds+1):

win = AB\_pruning(0, c\_player, -10000000, 100000000)

if win == -1:

scorpio +=1

results.append(f"Winner of Round {x}: Scorpion")

else:

subzero+= 1

results.append(f"Winner of Round {x}: Sub-Zero")

if c\_player == True:

c\_player = False

else:

c\_player = True

if scorpio > subzero:

T\_win = "Scorpian"

else:

T\_win = "Sub-Zero"

print(f"Game Winner: {T\_win}")

print(f"Total Rounds Played: {rounds}")

for p in results:

print(p)

initial\_player = f1.readline()

ini\_player = int(initial\_player)

game\_playing(ini\_player)

**Task no 02:**

f1 = open("20101539\_Umme Abira Azmary\_CSE422\_11\_Assignment03\_Summer2024\_InputFile2.txt", "r")

def Pac\_Minimax(node, depth, playerState, alpha, beta, leaf\_lst, darkMagic, c):

lst = []

if depth == 3:

return leaf\_lst[node]

if playerState == True:

emVal = -1000000000000 # should be (-infinity)

var = 0

for x in range(2):

val = Pac\_Minimax(2\*node+x, depth +1, False, alpha, beta, leaf\_lst, darkMagic, c)

emVal = max(emVal, val)

if depth == 0:

if emVal == val:

var = x

alpha = max(alpha, val)

if beta <= alpha:

break

if depth == 0:

return (emVal, var)

return emVal

else:

elVal = 1000000000000

for y in range(2):

val = Pac\_Minimax(2\*node+y, depth +1, True, alpha, beta, leaf\_lst, darkMagic, c)

lst.append(val)

elVal = min(elVal, val)

beta = min(beta, val)

if beta <= alpha:

break

if darkMagic == True:

elVal = max(lst)

return elVal

else:

return elVal

def pacman\_game(c):

leaf\_lst = [3, 6, 2, 3, 7, 1, 2, 0]

pacman\_magic = Pac\_Minimax(0, 0, True, -100000000000000, 1000000000000000, leaf\_lst, True, c)

#print(pacman\_magic)

pacman\_moves = Pac\_Minimax(0, 0, True, -100000000000000, 1000000000000000, leaf\_lst, False, c)

if pacman\_magic[0] - c > pacman\_moves[0]:

if pacman\_magic[1] == 0:

print(f"The new minimax value is {pacman\_magic[0]-c}. Pacman goes left and uses dark magic.")

else:

print(f"The new minimax value is {pacman\_magic[0]-c}. Pacman goes right and uses dark magic.")

else:

print(f"The minimax value is {pacman\_moves[0]}. Pacman does not use dark magic.")

temp = f1.readline()

c = int(temp)

pacman\_game(c)