**CSE422**

**LAB ASSIGNMENT 3**

**Mohammad Rakibul Hasan Mahin**

**20201220**

**==============================================================**

**TASK**

**==============================================================**

'''

CONSTRAINTS

1. There is going to be 4 levels

2. Level 0 -> Max -> Optimus Prime

3. Level 1 -> Min -> Megatron

4. Level 2 -> Max -> Optimus Prime

5. Level 3 -> Min -> Megatron

\*Any digit with 0 will get converted to 8

Minimum Point = 4th index of the ID

Maximum Point = Reverse last 2 digit and multiply with 1.5

Target = Reverse last 2 digit

Number of Shuffle = 3rd index of the ID

My ID: 20201220

28281228

Minimum Point = 1

Maximum Point = 82 \* 1.5 = 123

Target = 82

Shuffle = 8

'''

import random

import math

my\_id = input("Please Enter Your ID: ").replace('0','8')

# In this part we will extract all the information from the user input

min\_point = int(my\_id[4])

target\_point = my\_id[-2:]

target\_point = int(target\_point[::-1])

max\_point = math.ceil(target\_point \* 1.5)

shuffle\_number = int(my\_id[3])

# print(f"ID: {my\_id}\nMin Point: {min\_point}\nMax Point: {max\_point}\nTarget Point: {target\_point}\nShuffle Number: {shuffle\_number}")

# Initializing alpha as -ve infinity and beta as positive infinity

alpha, beta = float('-inf'), float('+inf')

# This function will calculate the total number of wins by Optimus Prime

def calculate\_number\_of\_wins\_for\_optumus\_prime(score\_list: int, target: int) -> int:

win\_count = 0

for i in score\_list:

if i >= target:

win\_count += 1

return win\_count

# This function compares two values and return the max

def max\_compare(val1: int, val2: int) -> int:

if val1 > val2:

return val1

else:

return val2

# This fucntion compares two values and return the min

def min\_compare(val1: int, val2: int) -> int:

if val1 < val2:

return val1

else:

return val2

# This is the main function that does alpha beta prunning on min-max algorithm

def alpha\_beta\_pruning(level: int, node: int, isMax: bool, point\_list: list, alpha: float, beta: float) -> int:

#As we are taking 8 leaf nodes,

#there won't be more than 4 level

#starting from 0 to 3

if level == 3:

return point\_list[node]

# This condition checks whether the turn is for max

if isMax:

best = alpha

# Here children are compared and values are updated

# Here we use max\_compare as it is the turn for max

# Only value of alpha is updated

for i in range(0,2):

curr\_score = alpha\_beta\_pruning(level+1, node\*2+i, False, point\_list, alpha, beta)

best = max\_compare(best, curr\_score)

alpha = max\_compare(best, alpha)

# This is the point where the prunning is done

if beta <= alpha:

break

return best

# This is when it is the turn for min

else:

best = beta

# Here children are compared and values are updated

# Here we use min\_compare as it is the turn for min

# Only value of beta is updated

for i in range(0,2):

curr\_score = alpha\_beta\_pruning(level+1, node\*2+i, True, point\_list, alpha, beta)

best = min\_compare(best, curr\_score)

beta = min\_compare(best, beta)

# This is the point where the prunning is done

if beta <= alpha:

break

return best

#=========================================TASK-1 OUTPUT==============================================#

print("#============================================TASK-1 OUTPUT=================================================#")

# 8 random numbers starting from min\_point to max\_point will be generated for the leaf nodes

point\_list = [random.randint(min\_point, max\_point) for \_ in range(8)]

print(f"Generated 8 random points between the minimum and maximum point limits: {point\_list}")

print(f"Total points to win: {target\_point}")

# Now we will call our alpha\_beta\_pruning function to calculate the score

score = alpha\_beta\_pruning(0,0,True,point\_list,alpha,beta)

print(f"Achieved point by applying alpha-beta prunning = {score}")

# Now compare the score and display the message accordingly

print("The winner is Optimus Prime") if score >= target\_point else print("The winner is Megatron")

#=========================================TASK-2 OUTPUT==============================================#

print("#============================================TASK-2 OUTPUT=================================================#")

all\_scores = []

# Run the loops according to the shuffle\_number generated from the ID

for i in range(shuffle\_number):

# 8 random numbers starting from min\_point to max\_point will be generated for the leaf nodes

# This point list will be newly generated after every iteration

point\_list = [random.randint(min\_point, max\_point) for \_ in range(8)]

# Now we will call our alpha\_beta\_pruning function to calculate the score

# And then add it to our all\_score list

score = alpha\_beta\_pruning(0,0,True,point\_list,alpha,beta)

all\_scores.append(score)

print(f"After the shuffle:\nList of all points values from each shuffles:{all\_scores}")

# We are sorting the list to find the highest score at the end of the list

all\_scores.sort()

highest\_score = all\_scores[-1]

print(f"The maximum value of all shuffles: {highest\_score}")

# Finally we will call calculate\_number\_of\_wins\_for\_optumus\_prime function

# and calculate number of times Optimus Prime won

win\_count = calculate\_number\_of\_wins\_for\_optumus\_prime(all\_scores, target\_point)

print(f"Won {win\_count} times out of {shuffle\_number} number of shuffles")