import random

def alpha\_beta\_pruning\_test(bullet\_no,depth,alpha,beta,idx,maximizing\_player,leaf\_node):

global prund\_nodes

if depth==0:

prund\_nodes+=1

return leaf\_node[idx]

if maximizing\_player==True:

damage=float('-inf')

for i in range(bullet\_no):

damage=max(damage,alpha\_beta\_pruning\_test(bullet\_no,depth-1,alpha,beta,(idx\*2)+i,False,leaf\_node))

alpha=max(alpha,damage)

if alpha>=beta:

break

return damage

else:

damage=float('inf')

for i in range(bullet\_no):

damage=min(damage,alpha\_beta\_pruning\_test(bullet\_no,depth-1,alpha,beta,(idx\*2)+i,False,leaf\_node))

beta=min(beta,damage)

if alpha>=beta:

break

return damage

student\_id=input("Enter your student ID: ")

min\_hp,max\_hp=input("Enter the minimum and maximum negative HP: ").split()

min\_hp,max\_hp=int(min\_hp),int(max\_hp)

turns=int(student\_id[0]) #numbers of turns for attacker

#print(f"Turns: {turns}")

depth=turns\*2

#print(f"Depth: {depth}")

defender\_initial\_hp=int(student\_id[-2:][::-1])

#print(f"Initial HP : {defender\_initial\_hp}")

prund\_nodes=0

number\_of\_bullets=int(student\_id[2]) #branches

leaf\_nodes\_lst=random.sample(range(min\_hp,max\_hp+1),pow(number\_of\_bullets,depth))

#print(f"Leaf node: {leaf\_nodefs\_lst}")

damage\_points=alpha\_beta\_pruning\_test(number\_of\_bullets,depth,float('-inf'),float('inf'),0,True,leaf\_nodes\_lst)

leaf\_nodes=''

for i in range(len(leaf\_nodes\_lst)):

leaf\_nodes+=str(leaf\_nodes\_lst[i])+','

leaf\_nodes=leaf\_nodes[:-1]

#---------------------------------

print(f"Depth and Branch Ratio is : {depth}:{number\_of\_bullets}")

print(f"Terminal States (leaf node values) are: {leaf\_nodes}")

print(f"Left life(HP) of the defender after maximum damage caused by the attacker is: {defender\_initial\_hp-damage\_points}")

print(f"After Alpha-Beta Pruning Leaf Node Comparisons: {prund\_nodes}")

#-----------------------------------