import math, random

comparisons = 0

*def* alphaBetaPruning(*index*, *depth*, *alpha*, *beta*, *attacker*):

*global* comparisons, branch

if depth == 0: return tree[index]

*# For Max Position*

if attacker:

tree[index] = -math.inf *#Alpha = Negative Infinity*

for i in range(branch \* index + 1, (branch \* index + branch) + 1, 1):

value = alphaBetaPruning(i, depth - 1, alpha, beta, False)

tree[index] = max(tree[index], value)

alpha = max(alpha, value) *# Max --> Alpha Update*

if alpha >= beta: *# Pruning Condition*

break

return tree[index]

*# For Min Position*

else:

tree[index] = math.inf *# Beta = Positive Infinity*

for i in range(branch \* index + 1, (branch \* index + branch) + 1, 1):

value = alphaBetaPruning(i, depth - 1, alpha, beta, True)

tree[index] = min(tree[index], value)

if depth == 1: comparisons += 1

beta = min(beta, value) *# Min --> Beta Update*

if alpha >= beta: *# Pruning Condistion*

break

return tree[index]

*#-------------- Code Runner (Main Method) -------------*

id = input("Enter your student id: \n")

minDamage, maxDamage = map(*int*, input("Minimum and Maximum value for the range of negative HP: \n").split(' '))

tree = []

initial\_hp = *int*(id[:-3:-1])

branch = *int*(id[2])

depth = *int*(id[0]) \* 2

print('\nDepth and Branches ratio is' ,depth,':',branch,)

for i in range(depth):

nodes = pow(branch, i)

for x in range(nodes):

if i % 2 == 0: tree.append(-math.inf)

else: tree.append(math.inf)

leafNodeRandom = [random.randint(minDamage, maxDamage) for x in range(pow(branch, depth))]

print('Terminal States(Leaf Nodes) are', \*leafNodeRandom, *sep*=', ', *end*='.\n')

tree = tree + leafNodeRandom

alphaBetaPruning(0, depth, -math.inf, math.inf, True)

remaining\_hp = initial\_hp - tree[0]

print(*f*'Left life(HP) of the defender after maximum damage caused by the attacker is {remaining\_hp}')

print(*f*'After Alpha-Beta Pruning Leaf Node Comparisons {comparisons}')