```
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}")
#---------------------------------
```