```
import random

print("Input :")
id=input("1. Enter your student id: ")
value_range=input("2. Minimum and Maximum value for the range of
negative HP: ").split(' ')

rand_start=int(value_range[0])
rand_end=int(value_range[1])

turn=int(id[0])
brunch=int(id[2])
HP=int(id[7]+id[6])
row=(turn*2)+1
table=[[]]



# creating table
# initially with column value 0

for i   in  range(row):

    for j in range(brunch**i):

        table[i].append(0)

    # removing extra row from table

    if (i < row-1):

        table.append([])



#putting random values in the last row
```

```
lastRow=len(table)-1
lastRowLength=len(table[lastRow])

for i   in  range(lastRowLength):

    table[lastRow][i]=random.randint(rand_start, rand_end)



##################
#applying min-max
##################

level=lastRow

for i   in  range(turn*2):

    start=0

    #if odd number then find min
    #else find max

    if((i+1)%2!=0):

        for j   in  range(int((brunch**level)/brunch)):

            end=start+brunch-1

            temp=table[level][start:end+1]
            table[level-1][j]=min(temp)

            start=end+1

    else:

        for j   in  range(int((brunch**level)/brunch)):

            end=start+brunch-1
```

```
            temp=table[level][start:end+1]
            table[level-1][j]=max(temp)

            start=end+1

    level=level-1



####################################
# Leaf Node Comparisons calculation
####################################

leaf_nodes=table[lastRow]

alpha=min(leaf_nodes[:brunch])
comparison=brunch
index=brunch

for i   in  range(int((len(leaf_nodes)/brunch)-1)):

    iterator=0

    while  iterator<brunch:

        if alpha>leaf_nodes[index]:

            comparison=comparison+1
            index=index+brunch-iterator
            break

        else:

            comparison=comparison+1

        index=index+1
        iterator=iterator+1
```

```
    if  iterator==brunch:

        start=(i+1)*brunch
        alpha=min(leaf_nodes[start:index])



##################
# Output
##################

print()

depth=turn*2
print("Depth and Branches ratio is ",depth,":",brunch)

print("Terminal States (leaf node values) are ",table[lastRow])

leftLife =(HP-table[0][0])
print("Left life(HP) of the defender after maximum damage caused by
the attacker is ",leftLife)

print("After Alpha-Beta Pruning Leaf Node Comparisons",comparison)
```