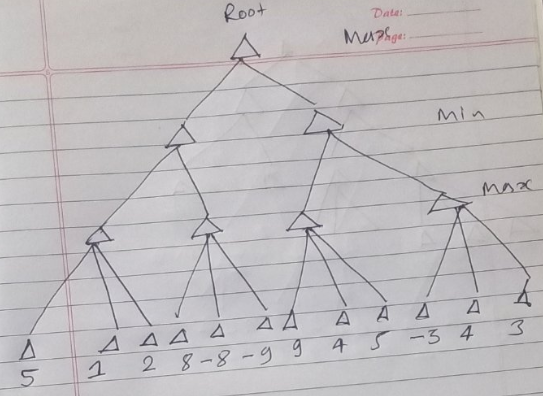
**Practical-no5: Perform the alpha beta pruning if exists.**



**Code:**

**Note: Here the value of alpha and beta are used as -15 and +15 respectively**

tree = [[[5, 1, 2], [8, -8, -9]], [[9, 4, 5], [-3, 4, 3]]]

root = 0

pruned = 0

def children(branch, depth, alpha, beta):

global tree

global root

global pruned

i = 0

for child in branch:

if type(child) is list:

(nalpha, nbeta) = children(child, depth + 1, alpha, beta)

if depth % 2 == 1:

beta = nalpha if nalpha < beta else beta

else:

alpha = nbeta if nbeta > alpha else alpha

branch[i] = alpha if depth % 2 == 0 else beta

i += 1

else:

if depth % 2 == 0 and alpha < child:

alpha = child

if depth % 2 == 1 and beta > child:

beta = child

if alpha >= beta:

pruned += 1

break

if depth == root:

tree = alpha if root == 0 else beta

return (alpha, beta)

def alphabeta(in\_tree=tree, start=root, upper=-15, lower=15):

global tree

global pruned

global root

(alpha, beta) = children(tree, start, upper, lower)

if \_\_name\_\_ == "\_\_main\_\_":

print ("(alpha, beta): ", alpha, beta)

print ("Result: ", tree)

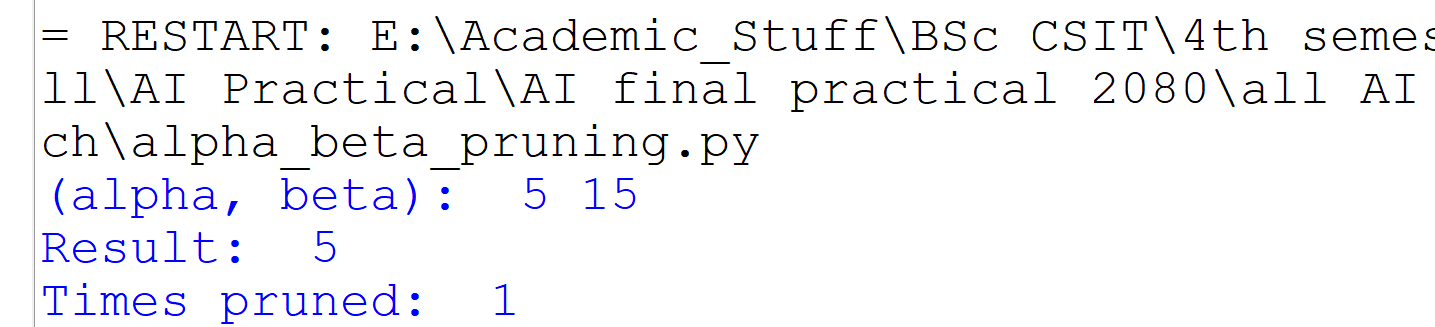
print ("Times pruned: ", pruned)

return (alpha, beta, tree, pruned)

if \_\_name\_\_ == "\_\_main\_\_":

alphabeta(None)

**Output:**



**Solve by hand written:**

