```
10. Alpha-Beta Pruning
#alpha beta pruning
class Node:
    def init (self, value=None, children=None):
        self.value = value
        self.children = children or []
def alpha beta (node, depth, alpha, beta, maximizing player):
    if depth == 0 or not node.children:
        return node.value
    if maximizing player:
        max eval = -float('inf')
        for child in node.children:
            eval = alpha beta(child, depth - 1, alpha, beta, False)
            max eval = max(max eval, eval)
            alpha = max(alpha, eval)
            print(f"Maximizing: alpha={alpha}, beta={beta}")
            if beta <= alpha:</pre>
                print(f"Pruning at node with value {node.value}")
        return max eval
    else:
        min eval = float('inf')
        for child in node.children:
            eval = alpha beta(child, depth - 1, alpha, beta, True)
            min eval = min(min eval, eval)
            beta = min(beta, eval)
            print(f"Minimizing: alpha={alpha}, beta={beta}")
            if beta <= alpha:</pre>
                print(f"Pruning at node with value {node.value}")
                break
        return min eval
root = Node(children=[
    Node (children=[Node (value=10), Node (value=9)]),
    Node (children=[Node (value=14), Node (value=18)]),
    Node (children=[Node (value=5), Node (value=4)]),
   Node (children=[Node (value=50), Node (value=3)])
1)
final value = alpha beta(root, 3, -float('inf'), float('inf'), True)
print(f"Final Value at MAX node: {final value}")
```

Output:

Minimizing: alpha=-inf, beta=10
Minimizing: alpha=-inf, beta=9
Maximizing: alpha=9, beta=inf
Minimizing: alpha=9, beta=14
Minimizing: alpha=9, beta=14
Maximizing: alpha=14, beta=inf
Minimizing: alpha=14, beta=5
Pruning at node with value None
Maximizing: alpha=14, beta=inf
Minimizing: alpha=14, beta=50
Minimizing: alpha=14, beta=3
Pruning at node with value None
Maximizing: alpha=14, beta=3
Fruning at node with value None
Maximizing: alpha=14, beta=inf
Final Value at MAX node: 14

(38) Lab -10 Analyse Alpha-Bela freeding mothed and simplement the same to wante offer Beter value generated to eductify find Value of max noch and Rubbiles Parened for a souver game true. Alposithin: RECOLUTION (KR. 940 des alpha seta (node, depth, alpha, beda, manimizero- player): if defith =0 on se termenal (morle): reduer Evaluate (noch) if monimizing - player: mon - Enal = - gloat ("inf") for child in node. children i Eval = alfha tota Cahelel, defth - 1, alpha, belo, False) max-enal = max (max- Enal. Enal) alpha = mon Calpha, Eval) if bela <= alpha: reduces mon Eval Elee: men- Eval = float (" inf") Jes cheld an node Chaldren: eval - alpho beta Coheld, depth -1, alpha, beta Toure boda = min Cheda, Eual) if beta <= alpha: reduces men - Eval.

pudfect:

minimizing: alpha = - inf, beta = 10

minimizing: alpha = - inf, beta = 9

mandowizing: alpha = 9, beta = 14

minimizing: alpha = 9, beta = 14

manimizing: alpha = 14, beta = 14

minimizing: alpha = 14, beta = 5

preening at node with value none

manimizing: alpha = 14, beta = inf

minimizing: alpha = 14, beta = 50

minimizing: alpha = 14, beta = 50

minimizing: alpha = 14, beta = 50

minimizing: alpha = 14, beta = 5

perentage at node with value none

manimizing: alpha = 14, beta = 50

minimizing: alpha = 14, beta = 5

perentage at node with value none

mandowizing: alpha = 14, beta = 14

Final value at MAX node: 14