SOURCE CODE : //Minimax without using alpha beta pruning

```
class TreeNode:
    def __init__(self, label=None):
        self.label = label
        self.children = []

def build_tree_from_input():
    print("Enter the tree nodes in a parent-children format (use '.' for no children):")
    root_label = input("Enter label for root node: ")
    root = TreeNode(root_label)
    queue = [root]

    while queue:
        current_node = queue.pop(0)

        children_input = input(f"Enter children for {current_node.label} (comma-separated labels or '.' if no children): ").strip()

        if children_input == '.':
            continue

        child_labels = list(map(str.strip, children_input.split(',')))
        for label in child_labels:
            if label == '.':
                continue
            new_child = TreeNode(label)
            current_node.children.append(new_child)
            queue.append(new_child)

    return root

def minimax(node, maximizingPlayer):
    if not node.children:
        return int(node.label)  # Convert the label to an integer if necessary

    if maximizingPlayer:
        best = float('-inf')
        for child in node.children:
            val = minimax(child, False)
            best = max(best, val)
        return best
    else:
        best = float('inf')
```

```
        for child in node.children:
            val = minimax(child, True)
            best = min(best, val)
        return best

# Example usage:
if __name__ == "__main__":
    # Build the tree dynamically from user input
    root = build_tree_from_input()

    if root is None:
        print("Empty tree!")
    else:
        # Perform basic minimax
        optimalValue = minimax(root, True)
        print("The optimal value is:", optimalValue)
```

OUTPUT :

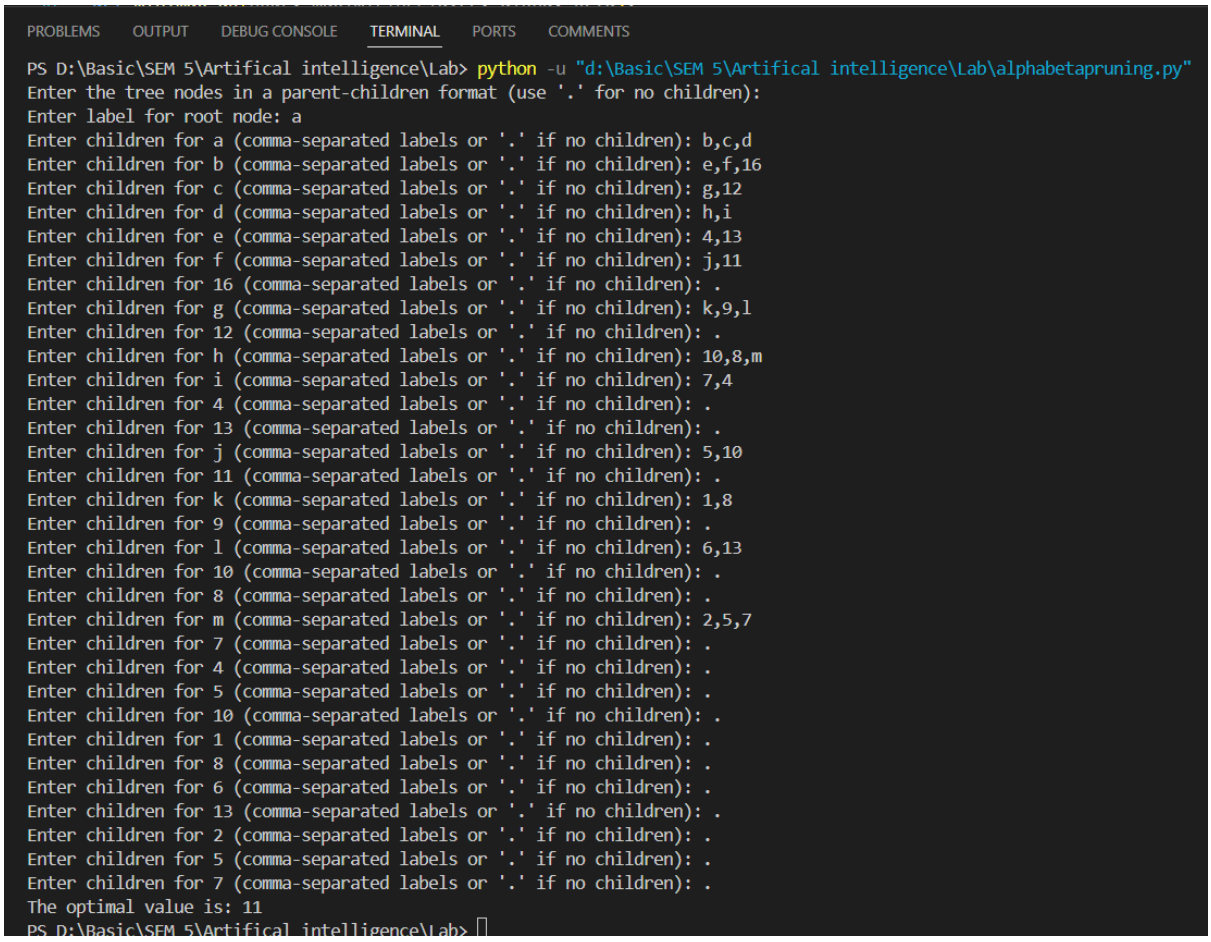
```
PS D:\Basic\SEM 5\Artifical intelligence\Lab> python -u "d:\Basic\SEM 5\Artifical intelligence\Lab\minmax.py"
Enter the tree nodes in a parent-children format (use '.' for no children):
Enter label for root node: a
Enter children for a (comma-separated labels or '.' if no children): b,c,d
Enter children for b (comma-separated labels or '.' if no children): e,f,16
Enter children for c (comma-separated labels or '.' if no children): g,12
Enter children for d (comma-separated labels or '.' if no children): h,i
Enter children for e (comma-separated labels or '.' if no children): 4,13
Enter children for f (comma-separated labels or '.' if no children): j,11
Enter children for 16 (comma-separated labels or '.' if no children): .
Enter children for g (comma-separated labels or '.' if no children): k,9,l
Enter children for 12 (comma-separated labels or '.' if no children): .
Enter children for h (comma-separated labels or '.' if no children): 10,8,m
Enter children for i (comma-separated labels or '.' if no children): 7,4
Enter children for 4 (comma-separated labels or '.' if no children): .
Enter children for 13 (comma-separated labels or '.' if no children): .
Enter children for j (comma-separated labels or '.' if no children): 5,10
Enter children for 11 (comma-separated labels or '.' if no children): .
Enter children for k (comma-separated labels or '.' if no children): 1,8
Enter children for 9 (comma-separated labels or '.' if no children): .
Enter children for l (comma-separated labels or '.' if no children): 6,12
Enter children for 10 (comma-separated labels or '.' if no children): .
Enter children for 8 (comma-separated labels or '.' if no children): .
Enter children for m (comma-separated labels or '.' if no children): 2,5,7
Enter children for 7 (comma-separated labels or '.' if no children): .
Enter children for 4 (comma-separated labels or '.' if no children): .
Enter children for 5 (comma-separated labels or '.' if no children): .
Enter children for 10 (comma-separated labels or '.' if no children): .
Enter children for 1 (comma-separated labels or '.' if no children): .
Enter children for 8 (comma-separated labels or '.' if no children): .
Enter children for 6 (comma-separated labels or '.' if no children): .
Enter children for 12 (comma-separated labels or '.' if no children): .
Enter children for 2 (comma-separated labels or '.' if no children): .
Enter children for 5 (comma-separated labels or '.' if no children): .
Enter children for 7 (comma-separated labels or '.' if no children): .
The optimal value is: 11
```

SOURCE CODE : // Minimax using alpha beta pruning

```
class TreeNode:
    def __init__(self, label=None):
        self.label = label
        self.children = []

def build_tree_from_input():
    print("Enter the tree nodes in a parent-children format (use '.' for no children):")
    root_label = input("Enter label for root node: ")
    root = TreeNode(root_label)
    queue = [root]

    while queue:
        current_node = queue.pop(0)

        children_input = input(f"Enter children for {current_node.label} (comma-separated labels or '.' if no children): ").strip()

        if children_input == '.':
            continue

        child_labels = list(map(str.strip, children_input.split(',')))
        for label in child_labels:
            if label == '.':
                continue
            new_child = TreeNode(label)
            current_node.children.append(new_child)
            queue.append(new_child)

    return root

def minimax_ab(node, maximizingPlayer, alpha, beta):
    if not node.children:
        return int(node.label)  # Convert the label to an integer if necessary

    if maximizingPlayer:
        best = float('-inf')
        for child in node.children:
            val = minimax_ab(child, False, alpha, beta)
            best = max(best, val)
            alpha = max(alpha, best)
            if beta <= alpha:
                break
```

```
        return best
    else:
        best = float('inf')
        for child in node.children:
            val = minimax_ab(child, True, alpha, beta)
            best = min(best, val)
            beta = min(beta, best)
            if beta <= alpha:
                break
        return best

# Example usage:
if __name__ == "__main__":
    # Build the tree dynamically from user input
    root = build_tree_from_input()

    if root is None:
        print("Empty tree!")
    else:
        # Perform minimax with alpha-beta pruning
        optimalValue = minimax_ab(root, True, float('-inf'), float('inf'))
        print("The optimal value is:", optimalValue)
```

OUTPUT :


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
PS D:\Basic\SEM 5\Artifical intelligence\Lab> python -u "d:\Basic\SEM 5\Artifical intelligence\Lab\alphabetapruning.py"
Enter the tree nodes in a parent-children format (use '.' for no children):
Enter label for root node: a
Enter children for a (comma-separated labels or '.' if no children): b,c,d
Enter children for b (comma-separated labels or '.' if no children): e,f,16
Enter children for c (comma-separated labels or '.' if no children): g,12
Enter children for d (comma-separated labels or '.' if no children): h,i
Enter children for e (comma-separated labels or '.' if no children): 4,13
Enter children for f (comma-separated labels or '.' if no children): j,11
Enter children for 16 (comma-separated labels or '.' if no children): .
Enter children for g (comma-separated labels or '.' if no children): k,9,l
Enter children for 12 (comma-separated labels or '.' if no children): .
Enter children for h (comma-separated labels or '.' if no children): 10,8,m
Enter children for i (comma-separated labels or '.' if no children): 7,4
Enter children for 4 (comma-separated labels or '.' if no children): .
Enter children for 13 (comma-separated labels or '.' if no children): .
Enter children for j (comma-separated labels or '.' if no children): 5,10
Enter children for 11 (comma-separated labels or '.' if no children): .
Enter children for k (comma-separated labels or '.' if no children): 1,8
Enter children for 9 (comma-separated labels or '.' if no children): .
Enter children for l (comma-separated labels or '.' if no children): 6,13
Enter children for 10 (comma-separated labels or '.' if no children): .
Enter children for 8 (comma-separated labels or '.' if no children): .
Enter children for m (comma-separated labels or '.' if no children): 2,5,7
Enter children for 7 (comma-separated labels or '.' if no children): .
Enter children for 4 (comma-separated labels or '.' if no children): .
Enter children for 5 (comma-separated labels or '.' if no children): .
Enter children for 10 (comma-separated labels or '.' if no children): .
Enter children for 1 (comma-separated labels or '.' if no children): .
Enter children for 8 (comma-separated labels or '.' if no children): .
Enter children for 6 (comma-separated labels or '.' if no children): .
Enter children for 13 (comma-separated labels or '.' if no children): .
Enter children for 2 (comma-separated labels or '.' if no children): .
Enter children for 5 (comma-separated labels or '.' if no children): .
Enter children for 7 (comma-separated labels or '.' if no children): .
The optimal value is: 11
PS D:\Basic\SEM 5\Artifical intelligence\Lab>