## #Alphabeta_pruning code :

```
def alphabeta(depth, index, alpha, beta, maximizingPlayer, values):
    # Base case: leaf node or depth limit reached
    if depth == 0 or index >= len(values):
        return values[index]

    if maximizingPlayer:
        best = float('-inf')
        # Left child
        val = alphabeta(depth-1, index*2, alpha, beta, False, values)
        best = max(best, val)
        alpha = max(alpha, best)
        if beta <= alpha:
            return best  # prune

        # Right child
        val = alphabeta(depth-1, index*2 + 1, alpha, beta, False, values)
        best = max(best, val)
        alpha = max(alpha, best)
        return best

    else:
        best = float('inf')
        # Left child
        val = alphabeta(depth-1, index*2, alpha, beta, True, values)
        best = min(best, val)
        beta = min(beta, best)
        if beta <= alpha:
            return best  # prune

        # Right child
        val = alphabeta(depth-1, index*2 + 1, alpha, beta, True, values)
        best = min(best, val)
        beta = min(beta, best)
        return best


# Example leaf values
```

```
values = [3, 5, 6, 9, 1, 2, 0, -1]
depth = 3
result = alphabeta(depth, 0, float('-inf'), float('inf'), True, values)

print("Optimal value (with Alpha-Beta Pruning):", result)
```