```python
import math

def alphabeta(depth, node_index, is_maximizing, values, alpha, beta, max_depth):
    if depth == max_depth:
        return values[node_index]

    if is_maximizing:
        best = -math.inf
        for i in range(2):
            val = alphabeta(depth + 1, node_index * 2 + i, False, values, alpha, beta, max_depth)
            best = max(best, val)
            alpha = max(alpha, best)

            print(f"Max node at depth {depth}: value={val}, alpha={alpha}, beta={beta}")

            if beta <= alpha:
                print("⛔ Pruned remaining branches (beta cutoff)")
                break
        return best
```

```
    else:
        best = math.inf
        for i in range(2):
            val = alphabeta(depth + 1, node_index * 2 + i, True, values, alpha, beta, max_depth)
            best = min(best, val)
            beta = min(beta, best)

            print(f"Min node at depth {depth}: value={val}, alpha={alpha}, beta={beta}")

            if beta <= alpha:
                print("⛔ Pruned remaining branches (alpha cutoff)")
                break
        return best

values = [3, 5, 6, 9, 1, 2, 0, -1]

print("Leaf Node Values:", values)
print("\nAlpha-Beta Pruning Process:\n")

result = alphabeta(0, 0, True, values, -math.inf, math.inf, 3)

print("\nOptimal value found:", result)

```

```
>>> %Run -c $EDITOR_CONTENT

  Leaf Node Values: [3, 5, 6, 9, 1, 2, 0, -1]

  Alpha-Beta Pruning Process:

  Max node at depth 2: value=3, alpha=3, beta=inf
  Max node at depth 2: value=5, alpha=5, beta=inf
  Min node at depth 1: value=5, alpha=-inf, beta=5
  Max node at depth 2: value=6, alpha=6, beta=5
  ⛔ Pruned remaining branches (beta cutoff)
  Min node at depth 1: value=6, alpha=-inf, beta=5
  Max node at depth 0: value=5, alpha=5, beta=inf
  Max node at depth 2: value=1, alpha=5, beta=inf
  Max node at depth 2: value=2, alpha=5, beta=inf
  Min node at depth 1: value=2, alpha=5, beta=2
  ⛔ Pruned remaining branches (alpha cutoff)
  Max node at depth 0: value=2, alpha=5, beta=inf

  Optimal value found: 5

>>>
```