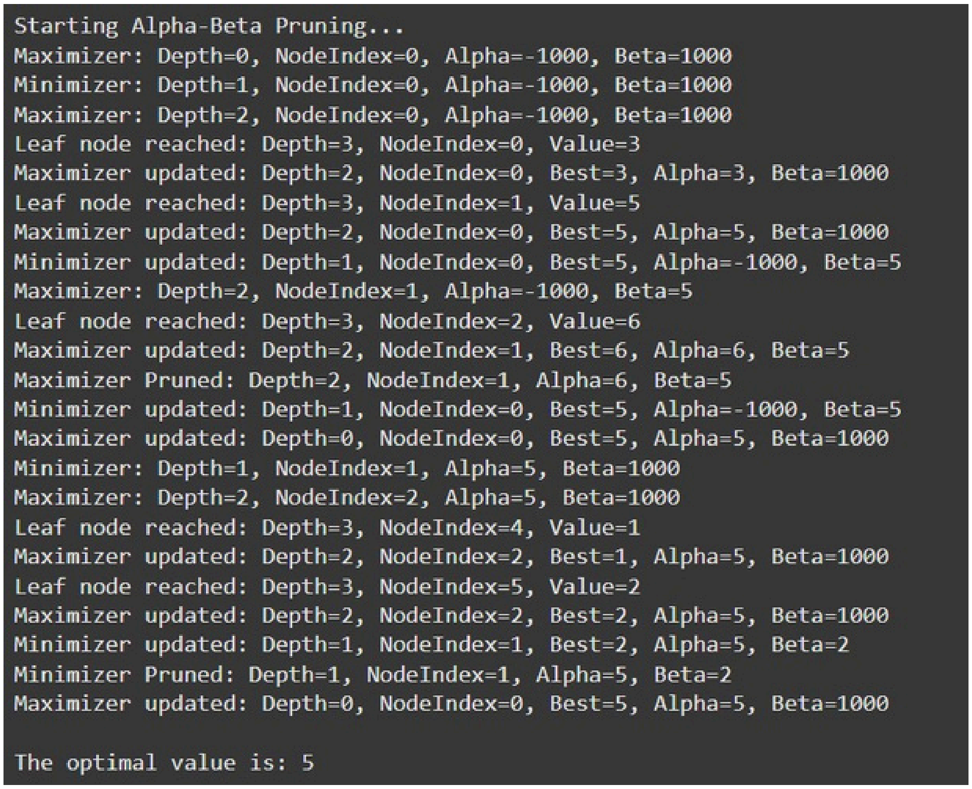
LAB-10 Alpha Beta Pruning

CODE:

```
# Python3 program to demonstrate
# working of Alpha-Beta Pruning with detailed step output

# Initial values of Alpha and Beta
MAX, MIN = 1000, -1000

# Returns optimal value for the current player
def minimax(depth, nodeIndex, maximizingPlayer, values, alpha, beta):
    # Terminating condition: leaf node is reached
    if depth == 3:
        print(f"Leaf node reached: Depth={depth}, NodeIndex={nodeIndex},
Value={values[nodeIndex]}")
        return values[nodeIndex]

    if maximizingPlayer:
        best = MIN
        print(f"Maximizer: Depth={depth}, NodeIndex={nodeIndex}, Alpha={alpha}, Beta={beta}")

        # Recur for left and right children
        for i in range(2):
            val = minimax(depth + 1, nodeIndex * 2 + i, False, values, alpha, beta)
            best = max(best, val)
            alpha = max(alpha, best)

            print(f"Maximizer updated: Depth={depth}, NodeIndex={nodeIndex}, Best={best}, Alpha={alpha}, Beta={beta}")

            # Alpha Beta Pruning
            if beta <= alpha:
                print(f"Maximizer Pruned: Depth={depth}, NodeIndex={nodeIndex}, Alpha={alpha}, Beta={beta}")
                break
        return best
    else:
        best = MAX
        print(f"Minimizer: Depth={depth}, NodeIndex={nodeIndex}, Alpha={alpha}, Beta={beta}")

        # Recur for left and right children
        for i in range(2):
            val = minimax(depth + 1, nodeIndex * 2 + i, True, values, alpha, beta)
            best = min(best, val)
            beta = min(beta, best)

            print(f"Minimizer updated: Depth={depth}, NodeIndex={nodeIndex}, Best={best}, Alpha={alpha}, Beta={beta}")

            # Alpha Beta Pruning
            if beta <= alpha:
                print(f"Minimizer Pruned: Depth={depth}, NodeIndex={nodeIndex}, Alpha={alpha}, Beta={beta}")
                break
        return best
```

```
# Driver Code
if _name_ == "_main_":
    values = [3, 5, 6, 9, 1, 2, 0, -1] # Leaf node values
    print("Starting Alpha-Beta Pruning...")
    optimal_value = minimax(0, 0, True, values, MIN, MAX)
    print(f"\nThe optimal value is: {optimal_value}")
```

OUTPUT:



```
Starting Alpha-Beta Pruning...
Maximizer: Depth=0, NodeIndex=0, Alpha=-1000, Beta=1000
Minimizer: Depth=1, NodeIndex=0, Alpha=-1000, Beta=1000
Maximizer: Depth=2, NodeIndex=0, Alpha=-1000, Beta=1000
Leaf node reached: Depth=3, NodeIndex=0, Value=3
Maximizer updated: Depth=2, NodeIndex=0, Best=3, Alpha=3, Beta=1000
Leaf node reached: Depth=3, NodeIndex=1, Value=5
Maximizer updated: Depth=2, NodeIndex=0, Best=5, Alpha=5, Beta=1000
Minimizer updated: Depth=1, NodeIndex=0, Best=5, Alpha=-1000, Beta=5
Maximizer: Depth=2, NodeIndex=1, Alpha=-1000, Beta=5
Leaf node reached: Depth=3, NodeIndex=2, Value=6
Maximizer updated: Depth=2, NodeIndex=1, Best=6, Alpha=6, Beta=5
Maximizer Pruned: Depth=2, NodeIndex=1, Alpha=6, Beta=5
Minimizer updated: Depth=1, NodeIndex=0, Best=5, Alpha=-1000, Beta=5
Maximizer updated: Depth=0, NodeIndex=0, Best=5, Alpha=5, Beta=1000
Minimizer: Depth=1, NodeIndex=1, Alpha=5, Beta=1000
Maximizer: Depth=2, NodeIndex=2, Alpha=5, Beta=1000
Leaf node reached: Depth=3, NodeIndex=4, Value=1
Maximizer updated: Depth=2, NodeIndex=2, Best=1, Alpha=5, Beta=1000
Leaf node reached: Depth=3, NodeIndex=5, Value=2
Maximizer updated: Depth=2, NodeIndex=2, Best=2, Alpha=5, Beta=1000
Minimizer updated: Depth=1, NodeIndex=1, Best=2, Alpha=5, Beta=2
Minimizer Pruned: Depth=1, NodeIndex=1, Alpha=5, Beta=2
Maximizer updated: Depth=0, NodeIndex=0, Best=5, Alpha=5, Beta=1000

The optimal value is: 5
```