Sagar Bangari(1BM22CS231)

LAB – 10 Alpha Beta Pruning

Code:

```
import math
def minimax(node, depth, is_maximizing):

  # Base case: Leaf node
  if node['left'] is None and node['right'] is None:
    return node['value']

  # Recursive case
  if is_maximizing:
    best_value = -math.inf
    if node['left']:
      best_value = max(best_value, minimax(node['left'], depth + 1, False))
    if node['right']:
      best_value = max(best_value, minimax(node['right'], depth + 1, False))
    return best_value
  else:
    best_value = math.inf
    if node['left']:
      best_value = min(best_value, minimax(node['left'], depth + 1, True))
    if node['right']:
      best_value = min(best_value, minimax(node['right'], depth + 1, True))
    return best_value
```

```
# Example usage
decision_tree = {
    'value': 5,
    'left': {
        'value': 6,
        'left': {
            'value': 7,
            'left': {
                'value': 4,
                'left': None,
                'right': None
            },
            'right': {
                'value': 5,
                'left': None,
                'right': None
            }
        },
        'right': {
            'value': 3,
            'left': {
                'value': 6,
                'left': None,
                'right': None
            },
            'right': {
```

```
            'value': 9,
            'left': None,
            'right': None
        }
    }
},
'right': {
    'value': 8,
    'left': {
        'value': 7,
        'left': {
            'value': 6,
            'left': None,
            'right': None
        },
        'right': {
            'value': 9,
            'left': None,
            'right': None
        }
    },
    'right': {
        'value': 8,
        'left': {
            'value': 6,
            'left': None,
```

```
            'right': None
        },
        'right': None
    }
  }
}

# Find the best move for the maximizing player
best_value = minimax(decision_tree, 0, True)
print(f"The best value for the maximizing player is: {best_value}")
```

Output:

```
The best value for the maximizing player is: 6
```