```
def minmax(depth, node_index, maximizing_player, values):
      if depth == 3:
              return values[node_index]

      if maximizing_player:
              return max(minmax(depth + 1, node_index * 2, False, values),
                         minmax(depth + 1, node_index * 2 + 1, False, values))
      else:
         return min(minmax(depth + 1, node_index * 2, True, values),
                    minmax(depth + 1, node_index * 2 + 1, True, values))

values = [-1 ,4 ,2 ,6 ,-3 ,-5 ,0 ,7]
print("The optimal value is :", minmax(0 ,0,True,values))
```

```
The optimal value is : 4
```

```
import math

def alpha_beta_pruning(depth, node_index, alpha, beta, maximizing_player, values):
    if depth == 3:
         return values[node_index]
    if maximizing_player:
              max_eval = -math.inf
              for i in range(2):
                 eval = alpha_beta_pruning(depth + 1, node_index * 2 + i, alpha, beta, False, values)
                 max_eval = max(max_eval, eval)

                 alpha = max(alpha, eval)
                 if beta <= alpha:
                    break
              return max_eval
    else:
       min_eval = math.inf
       for i in range(2):
          eval = alpha_beta_pruning(depth + 1, node_index * 2 + i, alpha, beta, True, values)
          min_eval = min(min_eval, eval)
          beta = min(beta, eval)
          if beta <= alpha:
              break
       return min_eval

values = [-1, 4, 2, 6, -3, -5, 0, 7]
print("The optimal value is :", alpha_beta_pruning(0, 0, -math.inf, math.inf,True,values))
```

```
The optimal value is : 4
```