TASK:4

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

# Returns optimal value for current player
# (Initially called for root and maximizer)
def minimax(depth, nodeIndex, maximizingPlayer, values, alpha, beta):
  # Terminating condition. i.e. leaf node is reached
  if depth == 3:
    return values[nodeIndex]

  if maximizingPlayer:
    best = MIN
    # 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)

      # Alpha Beta Pruning
      if beta <= alpha:
        break

    return best
  else:
    best = MAX
    # 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)
```

```
        # Alpha Beta Pruning
        if beta <= alpha:
            break

    return best

# Driver Code
if _name_ == "_main_":
    values = [3, 5, 6, 9, 1, 2, 0, -1]
    print("The optimal value is:", minimax(0, 0, True, values, MIN, MAX))
```

OUTPUT: