MinimaxInfo is a struct or class that stores the minimax value of a state *and* an action representing the best move from that state.

table is a hash table or dictionary that stores a mapping between game states and MinimaxInfo objects.

Terminal-Test(state) is a function that returns true if state is a terminal state (meaning the game has been won, lost, or drawn).

Utility(state) is a function that determines the numerical worth of a terminal state. Typically these values are positive for a MAX win and negative for a MIN win, with 0 meaning a draw.

Actions(state) is a function that returns all *legal* actions from a state.

Result(state, action) is a function that takes a state and an action and returns a new state (the successor state, or child state) that results from taking the action in the original state. This function assumes the action is a legal action from the state.

PlayerWhoMovesNext(state) is a function that takes a state and returns the player who moves next from that state. (This might just be a variable stored in the state object itself.)

```
function Minimax(state, table):
    // if we've already computed the minimax value for this state, don't do it again!
    if table contains state:
        return table[state].minimaxValue

    else if Terminal-Test(state):
        u = Utility(state)
        table[state] = MinimaxInfo(u, null)          // terminal states have no best move
        return u

    else if PlayerWhoMovesNext(state) == MAX:
        bestMinimaxSoFar = negative infinity
        bestMoveForState = null
        for each action a in Actions(state):
            childState = Result(state, a)
            minimaxOfChild = Minimax(childState, table)
            if minimaxOfChild > bestMinimaxSoFar:
                bestMinimaxSoFar = minimaxOfChild
                bestMoveForState = a
        table[state] = MinimaxInfo(bestMinimaxSoFar, bestMoveForState)
        return bestMinimaxSoFar

    else // PlayerWhoMovesNext(state) == MIN:
        bestMinimaxSoFar = infinity
        bestMoveForState = null
        for each action a in Actions(state):
            childState = Result(state, a)
            minimaxOfChild = Minimax(childState, table)
            if minimaxOfChild < bestMinimaxSoFar:
                bestMinimaxSoFar = minimaxOfChild
                bestMoveForState = a
        table[state] = MinimaxInfo(bestMinimaxSoFar, bestMoveForState)
        return bestMinimaxSoFar
```