Aufgabe 02)

Ausgabe meines Programmes:

Anzahl Knoten im Baum: 549946

Ergebnis Minimax: 0, besuchte Knoten: 549946

Ergebnis Pruning: 0, besuchte Knoten: 18297

Aufgabe 03)

Meine Bewertungslogik gibt den Gewinner wieder: -1 Min hat gewonnen, 1 Max hat gewonnen, 0 unentschieden.
Um die Funktion zu vereinfachen, könnte man den Spielerwechsel mathematisch lösen: durch summieren. Der Aufruf wäre dann kein Wechsel mehr, sondern ein rekursiver Aufruf.

Das wäre dann in Pseudo Code ausgedrückt:

```
Def minimax(state)
    if Terminal-Test(state): return Utility

    V = - unendlich

    For(a, s) in Succcessors(state):

        v = v + (minimax(s))

    Return v
```

Dann würde die -1 die +1 immer ausgleichen. Aber da keine Entscheidung stattfindet, sondern verrechnen, könnte auch ein anderer Wert als 1, -1 oder 0 auftauchen. (Ich bin grad meinen Baum durchgegangen). Mein nächster Gedanke war ein Vorzeichenwechsel, aber der hat dieselben Probleme: es ist ja ein Nullsummenspiel, es müsste ja 0 herauskommen.
Man müsste dann in der Funktion prüfen, welcher Spieler gerade dran ist, um zu entscheiden, welcher Wert genommen wird. Das wäre dann eine Funktion, aber die Spieler werden wieder unterschieden:

```
Def minimax(state)
    if Terminal-Test(state): return Utility

    V = - unendlich

    For(a, s) in Succcessors(state):

        wenn SpielerMin
        v = min (v, minimax(s))

        wenn SpielerMax
        v = max (v, minimax(s))

    Return v
```

Das sähe im Baum aber gleich aus, ob ich minimax oder zwei Funktionen nutze: