1. (6 Punkte) Durch die Dictionaries `nxt` und `blatt` ist ein Spielbaum mit der Wurzel `a` (Max-Knoten) gegeben.

```
nxt = {'a':list('bcd'),'b':list('efg'),'c':list('hij'),'d':list('klm')}
blatt = {'e':3,'f':12,'g':8,'h':2,'i':4,'j':6,'k':14,'l':2,'m':5}
```

   (a) Zeichne den Spielbaum und kennzeichne die Max- und Min-Ebenen.

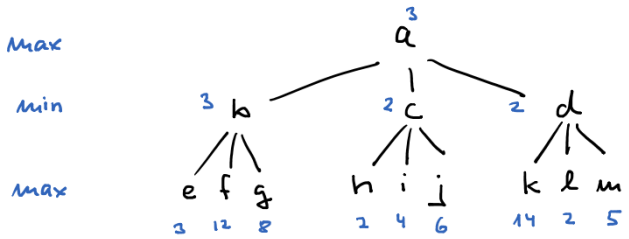
   (b) Gib an, in welcher Reihenfolge die Knoten ihre Werte erhalten. Gib dazu die Knotennamen zusammen mit den errechneten Werten an (Blätter müssen nicht aufgezählt werden).

   (c) Welches ist der beste Zug für `a`?

   (d) In welcher Reihenfolge besucht der Alpha-Beta-Algorithmus die Blätter? Notiere ein #, wenn ein pruning erfolgt, auch wenn danach kein Kind mehr übrig ist.

Lösung:

(a) Spielbaum mit den Ebenen und den Werten der Knoten:



(b) b:3 c:2 d:2 a:3

(c) b

(d) e f g h # k l #

2. (6 Punkte) Durch die Dictionaries `nxt` und `blatt` ist ein Spielbaum mit der Wurzel `a` (Max-Knoten) gegeben.

```
nxt = {'a':list('bc'),'b':list('de'),'c':list('fg'),'d':list('hi'),
    'e':list('jk'),'f':list('lm'),'g':list('no')}
blatt = {'h':5,'i':8,'j':9,'k':4,'l':7,'m':6,'n':5,'o':3}
```

(a) Zeichne den Spielbaum und kennzeichne die Max- und Min-Ebenen.

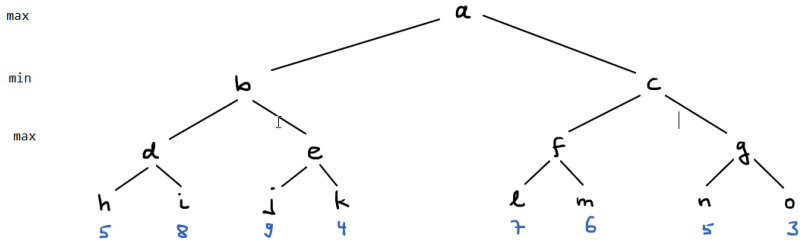
(b) Gib an, in welcher Reihenfolge die Knoten ihre Werte erhalten. Gib dazu die Knotennamen zusammen mit den errechneten Werten an (Blätter müssen nicht aufgezählt werden).

(c) Welches ist der beste Zug für `a`?

(d) In welcher Reihenfolge besucht der Alpha-Beta-Algorithmus die Blätter? Notiere ein #, wenn ein pruning erfolgt, auch wenn danach kein Kind mehr übrig ist.

Lösung:

(a) Spielbaum mit den Ebenen und den Werten der Knoten:



(b) d:8 e:9 b:8 f:7 g:5 c:5 a:8

(c) b

(d) h i j # l m #

3. (4 Punkte) Bei Tic-Tac-Toe sind die Felder so nummeriert:

```
0 1 2        x o x
3 4 5        . o .
6 7 8        x . o
```

Rechts steht die Stellung `[0, 1, 2, 4, 6, 8]`, x (Max) ist am Zug. Die Bewertung ist 1, wenn x gewonnen hat, -1, wenn o gewonnen hat, und 0 bei unentschieden.

(a) Zeichne den vollständigen Spielbaum ab dieser Stellung und bewerte alle Knoten.

(b) Welchen Zug wählt x? Was passiert bei den anderen beiden Zügen?

Lösung:

(a) Eingerückt stehen die Antworten auf den Zug darüber.

```
x auf 3: Wert 1 (Ende)
x auf 5: Wert -1
    o auf 3: Wert 0
        x auf 7: Wert 0 (Ende)
    o auf 7: Wert -1 (Ende)
x auf 7: Wert 0
    o auf 3: Wert 0
        x auf 5: Wert 0 (Ende)
    o auf 5: Wert 1
        x auf 3: Wert 1 (Ende)
```

Die Stellung hat den Wert 1.

(b) x setzt auf 3 und gewinnt sofort (Spalte 0, 3, 6). Setzt x auf 5, gewinnt o mit 7 (Spalte 1, 4, 7). Setzt x auf 7, endet das Spiel unentschieden.

4. (4 Punkte) (a) Liefert der Alpha-Beta-Algorithmus manchmal einen anderen Wert oder einen anderen besten Zug als der MinMax-Algorithmus? Begründe.

(b) Wovon hängt es ab, wie viel der Alpha-Beta-Algorithmus abschneiden kann?

(c) Wozu dient die Bremse? Welche Anforderung stellt sie an die Funktion `evaluation`?

(d) Der MinMax-Algorithmus liefert für die leere Tic-Tac-Toe-Stellung den Wert 0. Was bedeutet das?

Lösung:

(a) Nein. Abgeschnitten werden nur Teilbäume, die das Ergebnis nicht mehr ändern können, weil der Spieler darüber schon eine bessere Möglichkeit hat. Wert und bester Zug bleiben gleich.

(b) Von der Reihenfolge der Züge. Wird der beste Zug zuerst untersucht, sind α und β früh gut, und es wird viel abgeschnitten.

(c) Bei großen Spielbäumen (Vier gewinnt, Schach) kann man nicht bis zum Spielende suchen. Die Bremse begrenzt die Suchtiefe. Dann muss `evaluation` auch unfertige Stellungen bewerten können, also eine Heuristik sein, die schätzt, wie gut die Stellung für Max ist.

(d) Wenn beide Spieler perfekt spielen, endet Tic-Tac-Toe unentschieden. Keiner kann den Sieg erzwingen.

5. (5 Punkte) Auf dem Tisch liegen n Streichhölzer. Zwei Spieler nehmen abwechselnd 1, 2 oder 3 Hölzer weg. Wer das letzte Holz nimmt, hat verloren. Max beginnt.

   Eine Stellung soll das Tupel `(n, am_zug)` sein, wobei `am_zug` entweder `'max'` oder `'min'` ist. Schreibe die Funktionen `nextstates`, `terminal_test` und `evaluation`, so dass man mit `maximize((n, 'max'))` aus dem Unterricht herausfinden kann, ob Max gewinnen kann. Die Bewertung soll 1 sein, wenn Max gewinnt, und -1, wenn Min gewinnt.

Lösung:

```
def nextstates(state):
    n, am_zug = state
    anderer = 'min' if am_zug == 'max' else 'max'
    return [(n - k, anderer) for k in [1, 2, 3] if k <= n]

def terminal_test(state):
    n, am_zug = state
    return n == 0

def evaluation(state):
    n, am_zug = state
    # wer jetzt am Zug ist, hat nicht das letzte Holz genommen
    if am_zug == 'max':
        return 1
    return -1

for n in range(1, 9):
    print(n, maximize((n, 'max')))
```

Ist niemand mehr am Zug, weil die Hölzer aufgebraucht sind, dann hat der andere Spieler das letzte Holz genommen und verloren. Die Ausgabe zeigt: Max verliert bei $n = 1, 5, \ldots$, also wenn n beim Teilen durch 4 den Rest 1 lässt. Sonst gewinnt Max.