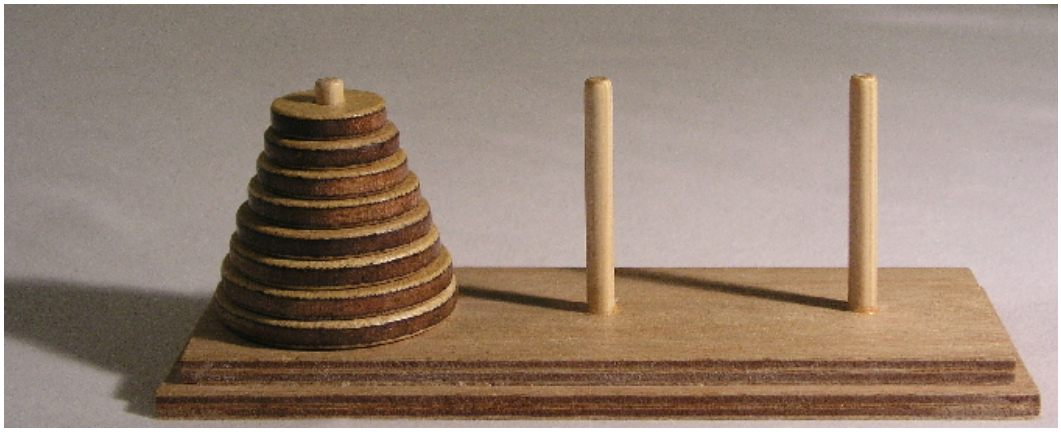
Aufgabe 1 (10 Punkte) [***]

Ein bekanntes Geduldsspiel sind die „Türme von Hanoi“. Beschreiben lässt sich dieses Spiel folgendermaßen:

> Das Spiel „Türme von Hanoi“ besteht aus drei Stäben a, b und c, auf die mehrere gelochte Scheiben gelegt werden, alle verschieden groß. Zu Beginn liegen alle Scheiben auf Stab a, der Größe nach geordnet, mit der größten Scheibe unten und der kleinsten oben. Ziel des Spiels ist es, den kompletten Scheiben-Stapel von a nach c zu versetzen.



> Bei jedem Zug darf die oberste Scheibe eines beliebigen Stabes auf einen der beiden anderen Stäbe gelegt werden, vorausgesetzt, dort liegt nicht schon eine kleinere Scheibe. Folglich sind zu jedem Zeitpunkt des Spieles die Scheiben auf jedem Feld der Größe nach geordnet.
>
> Für drei Scheiben sieht der korrekte Ablauf folgendermaßen aus:
>
> 1. Bewege oberste Scheibe von a nach c
> 2. Bewege oberste Scheibe von a nach b
> 3. Bewege oberste Scheibe von c nach b
> 4. Bewege oberste Scheibe von a nach c
> 5. Bewege oberste Scheibe von b nach a
> 6. Bewege oberste Scheibe von b nach c
> 7. Bewege oberste Scheibe von a nach c
>
> Eine interaktive Darstellung des Spieles für beliebige Anzahl von Scheiben finden Sie im Internet auch unter http://www.mathematik.ch/spiele/hanoi_mit_grafik/

Schreiben Sie eine Prozedur, die als Parameter die Anzahl `n` der Scheiben bekommt, die zu Beginn des Spieles auf Stab `a` liegen:

```
(define (tuerme-von-hanoi n)
                  ...)
```

Als Ausgabe soll die Prozedur eine Liste der durchgeführten Züge liefern. Jedes Element der Liste soll dabei ein Paar sein, dessen Linksanteil angibt, von welchem Stab die oberste Scheibe entfernt und dessen Rechtsanteil den Stab angibt auf den diese Schreibe gelegt wurde.

Beispiele:

```
(tuerme-von-hanoi 3)  →  ((a . c) (a . b) (c . b) (a . c) (b . a)
                         (b . c) (a . c))
```

Aufgabe 2 (6 Punkte) [**]

Schreiben Sie eine Prozedur, die eine Liste entgegennimmt und als Ergebnis die Elemente der Eingabe auf zwei Listen aufteilt. Dabei soll das erste Element in die erste Liste einsortiert werden, das zweite Elemente in die zweite Liste, das dritte Element in die erste Liste u.s.w.

```
(define (liste-teilen eingabe)
                ...)
```

Die beiden Listen sollen dabei innerhalb einer neuen Liste zurückgegeben werden. Die erste Liste an erster Stelle, die zweite Liste an zweiter Stelle.

Beispiele:

```
(liste-teilen '(1 2 3 4 5 6 7 8 9))      → ((1 3 5 7 9) (2 4 6 8))
(liste-teilen '(1 2 3 4 5 6 7 8 9 10)) → ((1 3 5 7 9) (2 4 6 8 10))
(liste-teilen '())                       → (() ())
```

Aufgabe 3 (5 Punkte) [*]

Schreiben Sie eine Prozedur, die das Ergebnis aus Aufgabe 42 nimmt und die beiden Listen wieder in einer Liste zusammenführt.

```
(define (listen-verschmelzen eingabe)
                ...)
```

Am folgenden Beispiel erkennt man, dass die Ergebnisliste aus dem ersten Element der ersten Liste, dem ersten Element der zweiten Liste, dem zweiten Element der ersten Liste, dem zweiten Element der zweiten Liste, dem dritten Element der ersten Liste u.s.w. besteht:

Beispiele:

```
(listen-verschmelzen '((1 3 5 7 9) (2 4 6 8)))      → (1 2 3 4 5 6 7 8 9)
(listen-verschmelzen '((1 3 5 7 9) (2 4 6 8 10))) → (1 2 3 4 5 6 7 8 9 10)
(listen-verschmelzen '(() ()))                       → ()
```

Aufgabe 4 (2 Punkte) [*]

Unter dem Hamming-Abstand zweier Binärzahlen versteht man die Anzahl der Binärstellen, an denen sie sich unterscheiden. Zum Beispiel ist der Hamming-Abstand von `10110101` und `01110100` 3. In Scheme lässt sich eine Binärzahl als eine Liste mit den Elementen 0 und 1 darstellen. Zum Beispiel wird dann die Binärzahl `10110101` als die Liste `(1 0 1 1 0 1 0 1)` dargestellt. Schreiben Sie eine Prozedur die den Hamming-Abstand zweier Binärzahlen ermittelt. Hierbei dürfen Sie davon ausgehen, dass die übergebenen Listen stets die gleiche Länge haben und nur die Zahlen 0 und 1 enthalten.

Beispiele:

```
(hamming '(1 0 1 1 0 1 0 1) '(0 1 1 1 0 1 0 0)) → 3
(hamming '(1 0 1) '(1 0 1))                       → 0
```

Tafelübung 9 [**]

Stellen Sie sich vor, Sie starten für jede Eingabe den Scheme-Interpreter neu und geben die folgenden Ausdrücke ein. Geben Sie für jede Eingabe das Ergebnis der Auswertung des angegebenen Ausdrucks an.

```
(define (was-ist-los liste1 liste2)
  (cond ((null? liste1) liste2)
        ((null? liste2) liste1)
        ((equal? (car liste1) (car liste2))
           (cons (car liste1)
                 (was-ist-los (cdr liste1) (cdr liste2))))
        (else (was-ist-los (cdr liste1) (cdr liste2)))))
```

Eingabe 1:

```
(was-ist-los '(1 3 4 5) '(2 3 4 6 7 8))
```

Eingabe 2:

```
(was-ist-los '(3 4 6 8) '(2 3 4 6 8 8))
```

Eingabe 3:

```
(was-ist-los '((1 2) (11 12) (21 22))
                     '((1 2 3) (11 12)))
```

Eingabe 4:

```
(was-ist-los '((1 2) (11 12) (21 22))
                     '(((1 2) (11 12) (21 22))))
```

Denken Sie an eine rechtzeitige Abgabe der Lösungen im Abgabesystem.