Funktionale und objektorientierte Programmierkonzepte

Florian Kadner und Lukas Röhrig
(Gesamtleitung: Prof. Karsten Weihe)

Wintersemester 18/19
v1.0

Übungsblatt 1

Themen:	Grundlagen, Listen und Strukturen
Relevante Folien:	Funktionale Abstraktion
Abgabe der Hausübung:	02.11.2018 bis 23:55 Uhr

V Vorbereitende Übungen

Denken Sie an Vertrag und drei Tests bei **jeder** Funktion.

V1 Temperaturumrechnung ☆ ☆ ☆

Im angloamerikanischen Maßsystem wird die Temperatur nicht wie hierzulande in Grad Celsius gemessen, sondern in Grad Fahrenheit. Da Sie damit nichts anfangen können, wollen Sie sich nun eine Funktion `fahr->cel` definieren, welche die aktuelle Temperatur in Grad Fahrenheit übergeben bekommt und in Grad Celsius umwandelt.

Hinweis: Für eine gegebene Temperatur T_F in Grad Fahrenheit berechnet sich die dazugehörige Temperatur T_C in Grad Celsius über den folgenden Zusammenhang:

$$T_C = (T_F - 32) \cdot \frac{5}{9}$$

V2 Volumen eines Tetraeders ★ ☆ ☆

Das Tetraeder ist ein Körper mit vier dreieckigen Seitenflächen. Sein Volumen berechnet sich über die Formel $V(a) = \frac{\sqrt{2}}{12}a^3$, wobei a hier die Länge einer Kante ist. Sie sollen in dieser Aufgabe nun eine Funktion `tetrahedron-volume` schreiben, die für eine übergebene Kantenlänge a das Volumen des dazugehörigen Tetraeders zurückgibt. Gehen Sie dazu in folgenden Schritten vor:

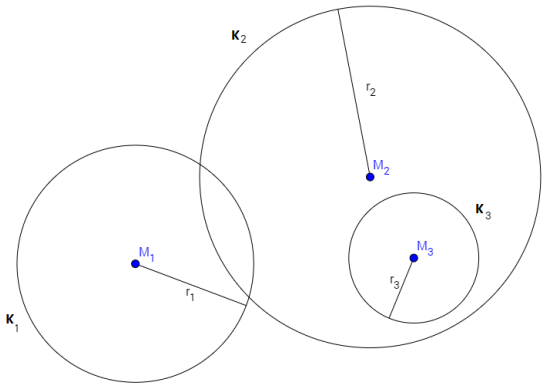
1. Definieren Sie eine Funktion `pow3`, welche einen Parameter x bekommt, und den Wert x^3 zurückgibt. Erstellen Sie außerdem eine Konstante k, mit dem Wert $\frac{\sqrt{2}}{12}$.
2. Nutzen Sie die zwei vorherigen Schritte, um nun die Funktion `tetrahedron-volume` zu definieren, welche nur einen Parameter `a` bekommt.

V3 Relative Lage zweier Kreise zueinander ★ ☆ ☆

Wir wollen die Lage zweier Kreise zueinander bestimmen. Definieren Sie dazu eine Prozedur `circles-position`, welche die Zahlen x_1, y_1, r_1, x_2, y_2, r_2 in dieser Reihenfolge entgegennimmt und einen String zurückgibt, welche die Lage der beiden Kreise zueinander beschreibt. Dabei sind x und y die Koordinaten eines Kreismittelpunktes und r der Radius des Kreises.

Zurückgegeben werden soll `"Intersect"` bei einem Schnitt der beiden Kreise, `"External"` bei keinerlei Überlappung oder `"Interior"` wenn einer der beiden Kreise vollständig im anderen liegt. Eine Berührung der beiden Kreise, egal ob von innen oder von außen, soll als Schnitt der beiden Kreise erkannt werden.

Zum besseren Verständnis finden Sie im Anschluss an die Aufgabe ein Beispiel und die mathematische Konkretisierung des Sachverhalts (dabei steht d für den Abstand der Mittelpunkte). Bei den Kreisen K_1 und K_2 im Beispiel sollte `"Intersect"`, bei den Kreisen K_1 und K_3 sollte `"External"` und bei den Kreisen K_2 und K_3 sollte `"Interior"` zurückgegeben werden.



$$\text{circles-position} = \begin{cases} \text{Interior} & \text{if } d < |r_1 - r_2| \\ \text{External} & \text{if } r_1 + r_2 < d \\ \text{Intersect} & \text{else} \end{cases}$$

V4 Euklidischer Algorithmus ★ ☆ ☆

In der folgenden Aufgabe soll das Konzept der Rekursion verinnerlicht werden. Dazu werfen wir einen Blick auf eine Version des euklidischen Algorithmus, welcher Ihnen vielleicht schon aus der Mathematik bekannt ist. Für zwei natürliche Zahlen a und b lässt sich mit ihm der größte gemeinsame Teiler der beiden Zahlen berechnen. Dabei geht der Algorithmus wie folgt vor:

Gilt $b = 0$ so wird a zurückgegeben, ist hingegen $a = 0$ so wird b zurückgegeben. Gilt $a > b$ so wird der Algorithmus mit einem neuen $\hat{a} = a - b$ und dem "alten" b aufgerufen. Im anderen Fall wird der Algorithmus mit dem "alten" a und einem neuen $\hat{b} = b - a$ aufgerufen. Definieren Sie eine Prozedur `(euclid a b)`, welche diese Version des euklidischen Algorithmus rekursiv umsetzt.

V5 Listenausdrücke auswerten ★ ☆ ☆

Teil A: Werden die folgenden Ausdrücke ohne Fehler durch Racket ausgeführt?

1. `(cons 1 (cons 2 (cons 3)))`
2. `(cons 1 (list 2 (list (list 3 + 4))))`
3. `(list (cons empty 1)(cons 2 empty)(cons 3 empty))`
4. `(first empty)`

Teil B: Liefern die folgenden Listenausdrücke dasselbe Ergebnis zurück?

1. `(cons 1 (cons 2 (cons 3 empty)))` und `(list 1 2 3 empty)`
2. `(cons (list "(list )")empty)` und `(list "list" empty)`
3. `(list 7 "*" 6 "=" 42)` und `(cons 7 (cons "*" (cons 6 (cons "=" (list 42)))))`

Teil C: Gehen Sie nun von folgendem Codeschnipsel aus:

```
(define A (list (cons 1 empty)(list 7) 9 ))
(define B (cons 42 (list "Hello" "world" "!")))
```

Was liefern die folgenden Aufrufe zurück?

1. `(first (rest A))`
2. `(rest (first A))`
3. `(append (first B)(rest (rest A))(first A))`

V6 Strukturausdrücke auswerten ★ ☆ ☆

Gegeben sei folgende Strukturdefinition:

```
(define-struct my-pair (one two))
```

Was liefern die folgenden Aufrufe zurück?

1. `(my-pair? (make-my-pair "a" "b"))`
2. `(make-my-pair 1 (make-my-pair 2 empty))`
3. `(* (my-pair-two (make-my-pair 1 2))(my-pair-one (make-my-pair 3 4)))`

V7 Listen in Strukturen ★ ☆ ☆

Gegeben ist ein Struct-Typ `abc` mit zwei Feldern `a` und `b`. Definieren Sie eine Funktion `foo` mit einem Parameter `p`. Falls `p` vom Typ `abc` und zudem der Wert im Feld `b` von `p` eine Liste ist, liefert `foo` eine Liste zurück, deren erstes Element der Wert von Feld `a` in `p` ist, und der Rest der zurückgelieferten Liste ist die Liste im Feld `b` von `p` (also eine Liste in der Liste). Andernfalls liefert foo einfach `false` zurück.

V8 Suche in Zahlenliste ★★☆

Definieren Sie eine Funktion `contains-x?`. Diese bekommt eine Liste und eine Zahl übergeben und liefert genau dann `true` zurück, wenn die übergebene Zahl mindestens einmal in der übergebenen Liste vorkommt.

V9 Duplikate in Zahlenliste ★★☆

Definieren Sie eine Prozedur `duplicates?` mit einem Parameter `lst`, die genau dann `true` zurückliefert, wenn eine Zahl mehr als einmal in `lst` vorkommt. *Hinweis*: Können Sie hier vielleicht Ihre Funktion aus Aufgabe V8 verwenden?

V10 Verschachtelte Listen ★★★

Definieren Sie eine Prozedur `duplicates-deep?` mit einem Parameter `deep-lst`. Es wird erwartet, dass `deep-lst` entweder eine Zahl oder eine Liste ist, deren Elemente wiederum Zahlen oder Listen sind usw. (Liste von Listen von Zahlen). Die Prozedur `duplicates-deep?` soll `true` oder `false` zurückliefern, und zwar `true` genau dann, wenn mindestens eine Zahl mehr als einmal vorkommt.

Hinweise: Schreiben Sie sich eine Hilfsmethode `collect` mit zwei Parametern `lst` und `oracle`. Der Sinn von `oracle` ist es dabei, alle bereits vorgekommenen Zahlen abzuspeichern. Machen Sie also aus der verschachtelten Liste wieder eine normale Liste mithilfe von `collect`. Die Funktion aus V9 kann Ihnen hier sehr hilfreich sein.

V11 Arithmetisches Mittel ★★★

Gegeben seien folgende Strukturdefinition:

```
(define-struct person (age sex))
(define-struct student (person id))
```

Eine Person hat ein Alter (als Zahl) und ein Geschlecht (als String). Ein Student wiederum besteht aus einer Person (vorheriges Struct) und einer Matrikelnummer (ein String).

Definieren Sie nun eine Funktion `mean-of-ages`. Diese bekommt eine Liste von Studenten übergeben und gibt das arithmetische Mittel ihrer Alter zurück.

Hinweis: Das arithmetische Mittel $\bar{x}$ berechnen Sie für n Alter $x_1, \ldots, x_n$ über:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i = \frac{x_1 + \cdots + x_n}{n}$$

H Erste Hausübung — Gesamt 6 Punkte
Binärbäume

Sie haben in Racket bereits die Datenstruktur *Liste* kennengelernt. In dieser ersten Hausübung betrachten Sie eine neue Datenstruktur - den *Binärbaum*. Nachfolgend wird es nur um Binärbäume mit der sogenannten Suchbaumeigenschaft gehen.

Werfen Sie einen Blick auf die Liste `(list 5 9 3 11 6 4 1)`. In Abbildung 1 wird diese Liste mithilfe eines Binärbaums dargestellt.

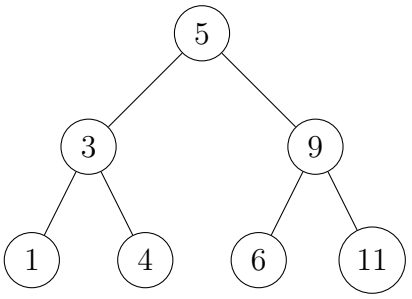


Abbildung 1: Ein Beispielbinärbaum mit Suchbaumeigenschaft

Der Baum besteht aus mehreren Teilen:

- Knoten des Baumes. Diese haben alle genau einen Vorgängerknoten und genau zwei Nachfolgeknoten. In unserem Beispiel: 3,9.
- Wurzel des Baumes. Der Knoten ohne Vorgänger und damit zentrales Element des Baumes. In unserem Beispiel: 5..
- Blätter des Baumes. Die Knoten ohne Nachfolger. In unserem Beispiel: 1,4,6,11.

Die Suchbaumeigenschaft ist die zentrale Eigenschaft. Sie sagt aus, dass in den linken Nachfolgern eines Knotens nur Werte stehen, die kleiner als der aktuelle Knoten sind. In den rechten Nachfolgeknoten stehen nur größere Werte. Machen Sie sich dies an dem Beispiel oben klar. Betrachten wir beispielhaft den Knoten mit dem Wert 9 so steht links der kleinere Nachfolgewert ($6 < 9$) und rechts der größere ($11 > 9$).

In der Hausübungsvorlage finden Sie bereits eine gegebene Definition der Baumknoten:

```
(define-struct binary-tree-node (value left right))
```

Jeder Knoten wird also als Struct realisiert, welches einen Wert `value` besitzt und außerdem die beiden Nachfolgeknoten `left` und `right`. Bei den Blättern des Baumes setzen wir die Nachfolgerknoten auf `empty`.

Sie werden in den nachfolgenden Teilaufgaben exakt angeleitet und begleitet. Dabei gehen wir Schritt für Schritt vor.

Für ein besseres Verständnis können Sie einen Blick auf diesen Binary Search Tree Visualizer werfen.

H1 Suche im Binärbaum 2 Punkte

Zu Beginn sollen Sie sich mit der wichtigsten Eigenschaft des Binärbaums auseinandersetzen, der Suche von Werten im Baum. Machen Sie sich nochmal bewusst, wie die Suche in einer gewöhnlichen Liste abläuft (vergleichen Sie dazu Aufgabe V8). Sie müssen hier jedes Element überprüfen und die Funktion immer wieder rekursiv auf dem Rest der Liste aufrufen. Im schlimmsten Fall müssen Sie die Funktion solange rekursiv aufrufen, bis Sie ganz am Ende der Liste angelangt sind. Sie benötigen in diesen *Worst Case* Fällen ganze n Vergleiche bei einer n-elementigen Eingabeliste. Bei einem Binärbaum mit Suchbaumeigenschaft werden hingegen deutlich weniger Aufrufe benötigt (näheres dazu lernen Sie in *Algorithmen und Datenstrukturen*).

Ergänzen Sie in dieser Aufgabe die Funktion `binary-tree-contains?`. Diese bekommt einen Wert als ersten und einen Binärbaum als zweiten Parameter übergeben. Die Funktion soll genau dann `true` zurückliefern, wenn der übergebene Wert im Binärbaum enthalten ist. Ist der übergebene Baum `empty`, so soll immer `false` zurückgegeben werden.

Hinweise:

- Der Binärbaum wird als `binary-tree-node`-Struct übergeben. Genauer gesagt wird die Wurzel des Baumes übergeben. Durch die Definition ihrer Nachfolger entstehen so verschachtelte Structs und insgesamt die Baumstruktur.
- Nutzen Sie die Suchbaumeigenschaft! Wenn Sie die Knoten mit dem übergebenen Wert vergleichen, können Sie leicht entscheiden, welchen Nachfolgeknoten Sie besuchen sollten.
- Überlegen Sie sich, wann Sie die Suche abbrechen. An welcher Stelle wissen Sie, dass der Baum den Wert nicht enthält?

In Abbildung 2 ist die (erfolglose) Suche nach dem Schlüsselwert 2 demonstriert. Der gelbe Knoten ist der aktuell besuchte, die roten Knoten sind bereits besuchte Knoten aus vergangenen Schritten. Machen Sie sich auch an diesem Beispiel nochmal klar, wie die Suchbaumeigenschaft funktioniert und wie Sie diese hier effektiv nutzen können.

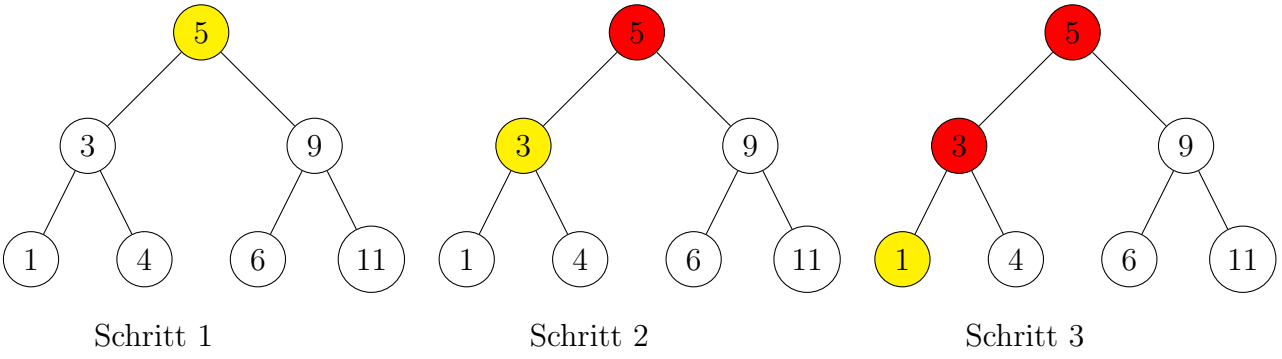


Abbildung 2: Erfolglose Suche nach dem Schlüsselwert 2

In der Vorlage sind bereits Binärbäume für Sie von uns angelegt worden. Sie können diese verwenden, um Ihre Implementierung zu testen. Außerdem sind schon von uns Testfälle vorgegeben. Wenn Sie diese nutzen wollen, kommentieren Sie diese einfach wieder ein.

H2 Einfügen in den Binärbaum 2 Punkte

Nun wollen wir neue Knoten in einen Binärbaum einfügen. Auch hier ist es wieder entscheidend, die Suchbaumeigenschaft auszunutzen. Als Beispiel nehmen wir wieder unseren altbekannten Baum und wollen den Wert 7 neu einfügen. In Abbildung 3 sind alle Knoten gelb markiert, die besucht werden, um die richtige Einfügeposition zu finden.

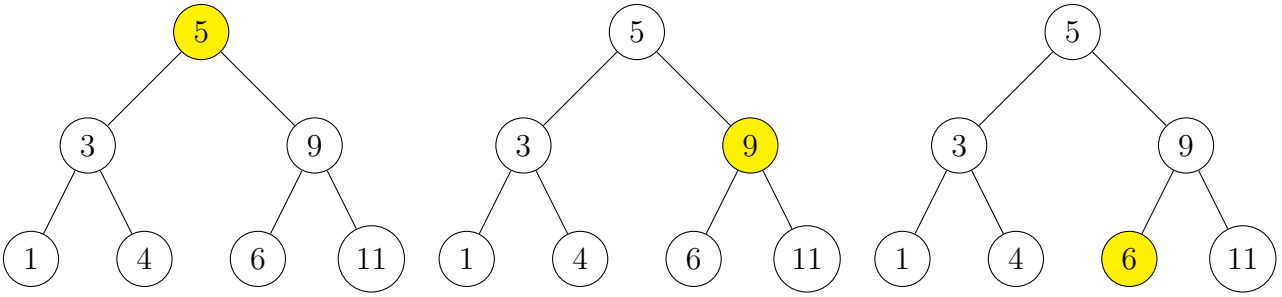


Abbildung 3: Finden der Position für den neuen Knoten mit Wert 7

Haben wir die richtige Stelle gefunden, so wird der neue Knoten eingefügt - hier in Abbildung 4 wieder gelb markiert.

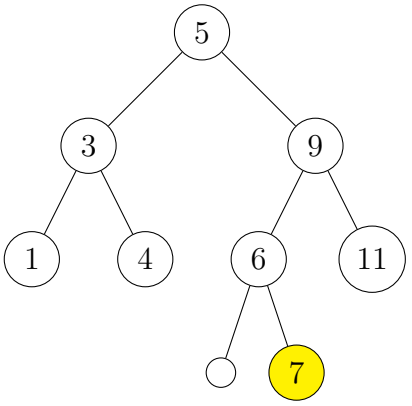


Abbildung 4: Der resultierende Baum nach dem Einfügen

Ergänzen Sie die Funktion `insert-into-binary-tree` in der Codevorlage. Diese erhält einen Wert und einen Binärbaum - wieder als Wurzelknoten-Struct - und soll diesen Wert korrekt einfügen. Soll ein Wert eingefügt werden, der bereits existiert, so soll der übergebene Baum ohne Modifizierung zurückgegeben werden.

Hinweis: Beachten Sie, dass der übergebene Binärbaum auch `empty` sein kann. In diesem Fall erstellen Sie nur die Wurzel des Baumes.

H3 Binärbaum aus Liste 1 Punkt

In dieser Aufgabe sollen Sie eine vorhandene Liste in die Binärbaum Datenstruktur umwandeln. Ergänzen Sie dazu die Funktion `binary-tree-from-list`, welche eine Liste von Zahlen übergeben bekommt. Die Funktion soll den aus der Liste resultierenden Baum zurückgeben, wenn die Zahlen der Liste in der gleichen Reihenfolge in einen leeren Binärbaum eingefügt werden. Ist die übergebene Liste leer, so soll die Funktion `false` zurückgeben.

Hinweis: Überlegen Sie sich, wie Sie bei dieser Aufgabe sinnvoll eine Hilfsmethode verwenden könnten.

H4 Duplikate im Binärbaum 1 Punkt

Im Normalfall kommen im Binärbaum keine Werte mehrfach vor. In dieser Aufgabe sollen Sie jedoch eine mögliche Behandlung für solche Duplikate implementieren. Ergänzen Sie die Funktion `insert-into-binary-tree-duplicates`. Diese bekommt wieder einen Binärbaum und einen Wert übergeben, der eingefügt werden soll. Die Besonderheit dieser Methode: Ist der Wert bereits im Binärbaum vorhanden, so soll das Feld `value` des `binary-tree-node`-Structs durch eine Liste ersetzt werden, in der der Wert n-mal vorkommt (bei n-maligem Einfügen in den Baum). Wird beispielsweise der Wert 7 jetzt 5 mal in den Baum eingefügt werden und befindet sich der Wert 7 an einem Blatt, so sieht das dazugehörige Struct folgendermaßen aus:

```
(make-binary-tree-node ((list 7 7 7 7 7) empty empty))
```

Hinweis: Sie können Ihre Funktion `insert-into-binary-tree` aus Übung H2 wiederverwenden und entsprechend modifizieren.