Lehrstuhl für Kognitive Modellierung
Prof. Dr. Martin V. Butz
Dr. Jan Kneissler

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Informatik II

Tafel-Übungsblatt 3

Ausgabe: Do, 08.05.2014 - **Abgabe: Do, 15.05.2014 - 23:59 Uhr**

Informationen: **Die Abgabe muss in elektronischer Form erfolgen!**
d.h. Ihre Bearbeitung muss in den angegebenen Formaten bis vor Mitternacht ins CIS System hochgeladen werden.

Aufgaben [32]

Notiz: Halten Sie sich beim Programmieren in Scheme an die eingeführte Konstruktionsanleitung:
a) Kurzbeschreibung als Kommentar (mittels: "; ..."");
b) Signatur (der "Vertrag" mittels: "(: ...)");
c) Testfälle (mittels "(check-within ...)" oder "(check-expect ...)" oder ...)
d) Prozedur Gerüst und Rumpf
Benutzen Sie Hilfsprozeduren, wenn Teilprobleme gelöst werden müssen!

3.1 Prozeduren mit Listen [23]

Schreiben Sie folgende rekursive Prozeduren, die auf Listen operieren.

a) Schreiben Sie eine Prozedur `last`, die auf einer beliebigen Liste operiert und nur das letzte Element dieser Liste zurückliefert. Ist die Liste leer, soll die Fehlermeldung `"Liste ist leer"` (`violation`) zurückgegeben werden. Benutzen Sie dabei **nicht** die eingebaute Prozedur `reverse`. [3]

b) Schreiben Sie eine Prozedur `elem?`, die eine Zahl und eine Liste von Zahlen erwartet und überprüft, ob diese Zahl ein Element der Liste ist. Beachten Sie dass die Signatur der Ausgabe von der Signatur `boolean` ist und verwenden Sie hier nicht `violation`. [3]

c) Schreiben Sie eine Prozedur `which-elem`, die eine Zahl und eine Liste von Zahlen erwartet und die Position in der Liste zurückliefert, an der diese Zahl das erste mal aufgefunden wurde. Geben Sie `-1` zurück, falls das Element nicht gefunden wurde. Das erste Element in der Liste steht an Position 0. Hinweis: Es ist sinnvoll sich eine Hilfs-Prozedur zu implementieren, die zusätzlich einen Zähler übergeben bekommt (anfangs 0) und in der die rekursive Suche nach der Zahl stattfindet. [5]

d) Schreiben Sie eine Prozedur `sorted?`, die eine Liste von natürlichen Zahlen und einen boolean erwartet und die Liste entweder auf aufsteigende Sortierung hin überprüft, wenn der Boolsche Wert wahr ist und auf absteigende Sortierung überprüft, wenn der Boolsche Wert falsch ist. Eine leere Liste ist per Definition immer sortiert. [5]

e) Schreiben Sie eine Prozedur `all-equal?`, welche eine Liste von Zahlen daraufhin prüft, ob alle Elemente gleich sind. Im Falle einer leeren Liste soll `#t` ausgegeben werden. [3]

f) Schreiben Sie eine Prozedur `all-not-equal?`, welche eine Liste von Zahlen daraufhin prüft, ob alle Elemente unterschiedliche sind. Nutzen Sie dafür die von Ihnen bereits implementierte Prozedur `elem?`. Im Falle einer leeren Liste soll `#t` ausgegeben werden. [4]

Abgabe: Programm `list-functions.rkt`

3.2 Spalte und Vereine Listen[9]

Schreiben Sie folgende Prozeduren auf Listen:

a) Schreiben Sie eine Prozedur `split-list`, die eine beliebige Liste konsumiert und die Elemente der Liste abwechselnd auf die beiden Ergebnislisten verteilt. Dabei soll `split-list` die Liste nur einmal ablaufen, Sie müssen also in einem Schritt zwei Elemente der Liste betrachten, sofern die Liste noch lang genug ist. Stellen Sie sicher, dass Ihre Implementierung auch für leere Listen korrekte Ergebnisse liefert!
Die Prozedur liefert als Ausgabe ein Paar von Listen. Konstruieren Sie dazu einen parametrischen Record-Typ `paar`. [4]

b) Schreiben Sie eine Prozedur `weave-lists`, die ein Paar mit zwei Listen konsumiert und eine Liste zurückgibt. Die Ergebnisliste soll die Elemente der beiden gegebenen Listen abwechselnd enthalten. [3]

c) Überprüfen Sie mit `check-property`, dass über `split-list` und `weave-lists` für beliebige Listen l die folgende Äquivalenz gilt (es ist ausreichend, wenn Sie über Listen von natürlichen Zahlen testen): [2]

$$\texttt{(weave-lists (split-list } l\texttt{))} \equiv l$$

Abgabe: Programm `split-weave-lists.rkt`