Informatik I

PD Dr. J.-G. Smaus
G. Röger, R. Mattmüller
Wintersemester 2010/2011

Universität Freiburg
Institut für Informatik

Übungsblatt 3

Abgabe: 2. Dezember 2010

Aufgabe 3.1 (Die Macht der Abstraktion, 2 + 2 + 1 Punkte)

Verwenden Sie für diese Aufgabe die Definitionen aus der Datei `zlisten.rkt` von der Übungsseite. Die Datei enthält neben den aus der Vorlesung bekannten Zahlenlisten noch eine Prozedur `(: ausgabe (zahlenliste -> string))`, die eine Zahlenliste in eine besser lesbare Repräsentation umwandelt.

(a) Schreiben Sie eine Prozedur `(: enum-rev (natural -> zahlenliste))`, die für eine natürliche Zahl `n` die Liste der Zahlen `n, n-a, ..., 1` liefert. Beispiele:

- `(enum-rev 2)` liefert die Liste `(kons 2 (kons 1 leer))`.
- `(enum-rev 0)` liefert die Liste `leer`.

Geben Sie an, welche Konstruktionsanleitung Sie verwendet haben, und schreiben Sie mindestens drei Testfälle.

(b) Betrachten wir anstelle der eingebauten Listen unsere Zahlenlisten, sieht die Konstruktionsanleitung für eine Prozedur mit Akkumulator folgendermaßen aus:

```
(: proc (zahlenliste -> τ))
(define proc
    (lambda (l)
        (proc-helper l initial)))

(: proc-helper (zahlenliste τ -> τ))
(define proc-helper
    (lambda (l acc)
        (if (empty? l)
            acc
            (proc-helper (rumpf l)
                (... (kopf l) ... acc ...)))))
```

Dabei muss `initial` ein neutrales, initiales „Zwischenergebnis“ sein, und `(... (kopf l) ... acc ...)` berechnet das nächste Zwischenergebnis.

Verwenden Sie diese Konstruktionsanleitung, um eine Prozedur `(: revert (zahlenliste -> zahlenliste))` zu schreiben, die zu einer Zahlenliste die umgekehrte Liste zurückgibt. Beispiele:

- `(revert (kons 3 (kons 5 (kons 2 leer))))` liefert die Liste `(kons 2 (kons 5 (kons 3 leer)))`.
- `(revert (kons 1 (kons 2 leer)))` liefert die Liste `(kons 2 (kons 1 leer))`.
- `(revert leer)` liefert `leer`.

Schreiben Sie mindestens drei Testfälle.

(c) Schreiben Sie eine Prozedur `(: enumerate (natural -> zahlenliste))`, die für eine natürliche Zahl `n` die Liste der Zahlen `1, 2, ..., n` liefert. Schreiben Sie mindestens drei Testfälle. Verwenden Sie die Prozeduren aus Teilaufgabe a und b.

Abgabe: elektronisch (in einer Datei `enumerate.rkt`) oder auf Papier.

Aufgabe 3.2 (Die Macht der Abstraktion, 1 + 1 Punkte)

(a) Beschreiben Sie den Unterschied zwischen den beiden Konstrukten `let` und `let*` in eigenen Worten.

(b) Geben Sie zur Demonstration zwei verschiedene Programme an, deren Code sich nur darin unterscheidet, dass das eine Programm `let` und das andere `let*` verwendet, und zeigen Sie dass die Programme unterschiedliche Ergebnisse berechnen. Verwenden Sie *nicht* das Beispiel aus der Vorlesung!

Abgabe: elektronisch in einer Textdatei (`let.txt`) oder auf Papier

Aufgabe 3.3 (3 Punkte)

In der Vorlesung wurden die Begriffe *Links-* und *Rechtseindeutigkeit*, *Reflexivität*, *Irreflexivität*, *Symmetrie*, *Antisymmetrie*, *Transitivität* und *Äquivalenz* eingeführt. Wir haben neun Beispielrelationen eingeführt und jeweils betrachtet, ob sie eine der Eigenschaften haben oder nicht.
Hierbei waren einige der Beispielrelationen keine konkreten Beispiele, sondern parametrisiert (also von einem nicht gegebenen A bzw. B abhängig):

- $A \times B$ (bzw. Spezialfall $A \times A$)
- $\emptyset \subseteq A \times B$ (bzw. Spezialfall $\emptyset \subseteq A \times A$)
- $I_A \subseteq A \times A$ mit $I_A = \{(a, a) \mid a \in A\}$

Hierbei kann es sein, dass die Behauptung aus der Vorlesung, die Beispielrelation habe Eigenschaft X (nicht), in gewissen Grenzfällen (A oder B leer oder einelementig) falsch ist. Gehen Sie die Vorlesungsfolien durch und benennen Sie alle diese Fälle, d.h. schreiben Sie etwas wie „Relation R hat Eigenschaft X, sofern ...“, wenn es in den Folien lediglich heißt „Relation R hat Eigenschaft X.“

Abgabe: elektronisch als eine mit LaTeX erstellte PDF-Datei (`relationen.pdf`) oder auf Papier.

Um für die Programmieraufgaben Punkte zu erhalten, folgen Sie den Konstruktionsanleitungen der Vorlesung, d. h.:

1. Geben Sie die Signatur an.
   Falls die Signatur fehlt, gibt die Aufgabe 0 Punkte.
2. Wählen Sie abhängig von der Signatur das richtige Funktionsgerüst aus.
3. Geben Sie Testfälle an.
4. Schreiben Sie den Funktionsrumpf. Dieser Schritt gliedert sich in weitere Unterschritte, die Sie entsprechend auswählen müssen.
   Sie erhalten für diesen Teil der Aufgabe nur Punkte, wenn Sie Schritte 1 bis 3 befolgt haben.

Die Übungsblätter müssen individuell bearbeitet werden. Gruppenabgaben sind nicht zulässig.