Grundlagen der Informatik 1
WS 2008/09

Prof. Mühlhäuser, Dr. Rößling, Melanie Hartmann, Daniel Schreiber
http://proffs.tk.informatik.tu-darmstadt.de/gdi1

Übung 2 27.10.2008

---

1 Mini Quiz

☐ Elemente einer Liste können von verschiedenen Datentypen sein.

☐ Rekursive Datentypen und Prozeduren brauchen zum Terminieren immer einen Rekursionsanker.

☐ Man kann mit `cons` Listen erzeugen, die man mit `list` nicht erzeugen kann.

☐ Strukturdefinitionen erzeugen automatisch Konstruktoren, Selektoren und Prädikate.

☐ Natürliche Zahlen sind abgeschlossen unter der Subtraktion.

2 Fragen

1. Beschreiben Sie mit eigenen Worten, was ein rekursiver Datentyp ist.
2. Erklären Sie wieso rekursive Datenstrukturen in der Praxis nicht unendlich groß werden.
3. Welche zwei Ansätze beim Design von Programmen mit Hilfe von Wunschlisten sind Ihnen bekannt? Diskutieren Sie Vor- und Nachteile des jeweiligen Ansatzes.
4. Erklären Sie mit eigenen Worten, was Abgeschlossenheit ist.

3 Strukturen (K)

Versuchen Sie bitte die Aufgaben erst ohne einen Rechner zu lösen.
Gegeben sei folgende Strukturdefinition:

```
(define-struct mypair (first second))
```

Werten Sie folgende Ausdrück aus und geben Sie das Ergebnis an.

1. `(make-mypair 'a 'b)`
2. `(mypair? (make-mypair 'a 'b))`
3. `(mypair? (list 'a 'b))`
4. `(make-mypair 1 (make-mypair 2 empty))`
5. `(* (mypair-second (make-mypair 1 2)) (mypair-first (make-mypair 3 4)))`

4 Listen (K)

Nehmen Sie für die folgenden Aufgaben das Sprachlevel „Anfänger mit Listen-Abkürzungen“ an. Versuchen Sie bitte die Aufgaben erst ohne einen Rechner zu lösen. Diese Aufgabe muss bearbeitet werden, um die Hausübung lösen zu können.

1. Welche der folgenden Ausdruckspaare werden zu äquivalenten Listen ausgewertet?
   a) `(cons 1 (cons 2 (cons 3 empty)))` und `(list 1 2 3 empty)`
   b) `(cons (list '()) empty)` und `(list 'list empty)`
   c) `(list 7 '* 6 '= 42)` und `(cons 7 (cons '* (cons 6 (cons '= (list 42)))))`
   d) `(cons 'A (list '(I)))` und `(list 'A (cons 'I empty))`
2. Werden folgende Ausdrücke fehlerfrei ausgewertet? Falls nicht, begründen Sie bitte was zum Fehler führt.
   a) `(cons 1 (cons 2 (cons 3)))`
   b) `(cons 1 (list 2 (list '(3 + 4))))`
   c) `(list (cons empty 1) (cons 2 empty) (cons 3 empty))`
3. In der Vorlesung haben Sie erfahren, wie man Listen in Kästchenschreibweise darstellt (T3.8). Stellen Sie folgende Listen in Kästchenschreibweise dar:
   a) `(define A (list (cons 1 empty) (list 7) 9))`
   b) `(define B (cons 42 (list 'Hello 'world '!)))`
4. Zu welchen Werten werden folgende Ausdrücke (A und B wie oben definiert) ausgewertet? Falls ein Auswertung nicht möglich ist, begründen Sie bitte.
   a) `(first (rest A))`
   b) `(rest (first A))`
   c) `(first empty)`
   d) `(append (first B) (rest (rest A)) (first A))`
5. Im folgenden sollen Sie rekursive Prozeduren auf Listen definieren. Vergessen Sie nicht, zuerst Vertrag, Beschreibung und ein Beispiel anzugeben.
   a) Schreiben Sie eine Prozedur `list-length: lst -> num`, die eine Liste als Parameter erhält und die Länge der Liste zurückliefert. (Hinweis: Die leere Liste enthält keine Elemente. Verwenden Sie Rekursion, um das Ergebnis für nicht-leere Listen zu berechnen.)
   b) Schreiben Sie eine Prozedur `member?: los symbol -> boolean`, die eine Liste von Symbolen und ein Symbol als Parameter erhält und zurückliefert, ob das Symbol in der Liste enthalten ist. (Hinweis: Die leere Liste enthält das Symbol sicher nicht. Verwenden Sie Rekursion, um das Ergebnis für nicht-leere Listen zu berechnen. Verwenden Sie die Prozedur `symbol=?: symbol symbol -> boolean`, um zu überprüfen ob zwei Symbole identisch sind.)

c) Schreiben Sie eine Prozedur `remove-duplicates: los -> los`, die eine Liste von Symbolen als Parameter erhält und alle Duplikate aus der Liste entfernt.
Beispiel: `(remove-duplicates '(a a b) -> '(a b)`.
Verwenden Sie Rekursion und betrachten Sie drei Fälle:

- Die leere Liste.
- Das erste Element der Liste kommt im Rest der Liste vor.
- Das erste Element kommt nicht im Rest der Liste vor.

Hausübung

Die Vorlagen für die Bearbeitung werden im GdI1-Portal bereitgestellt. Kommentieren Sie Ihren selbst erstellten Code. Die Hausübung muss bis zum Abgabedatum im GdI1-Portal abgegeben werden. Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Zu diesen gehört auch die strikte Verfolgung von Plagiarismus. Mit der Abgabe Ihrer Hausübung bestätigen Sie, dass Sie bzw. Ihre Gruppe alleiniger Autor des gesamten Materials sind. Falls Ihnen die Verwendung von Fremdmaterial gestattet war, so müssen Sie dessen Quellen deutlich zitieren.

Abgabe: Spätestens Fr, 07.11.08, 16:00.
Wichtig: Um die volle Punktzahl zu erlangen, müssen Sie jede Ihrer Prozeduren und Hilfsprozeduren mindestens mit Vertrag, Beschreibung und Beispiel kommentieren, sowie jeweils Testfälle angeben. Verwenden Sie als Sprachlevel für die gesamte Hausübung „Anfänger mit Listen-Abkürzungen“.

5 EMail Adresse

1. Tragen Sie eine gültige EMail Adresse im GdI Portal ein („Mein Konto“->Bearbeiten), sonst kann es sein, dass wichtige eMails Ihres Tutors Sie nicht erreichen!!

2. Lesen Sie entweder regelmäßig die EMails in ihrer RBG Mailbox oder richten Sie sich einen Mail forward ein. Sollten Sie Probleme damit haben, folgen Sie den Hinweisen hier: www.rbg.informatik.tu-darmstadt.de/index.php?id=560#forward.

6 Listen und Strukturen für Suchmaschinen (13P)

In dieser Aufgabe werden Teile der Funktionsweise einer Internet-Suchmaschine auf sehr einfache Art und Weise nachgebaut.
Wir beginnen mit der Definition einer Ähnlichkeitsfunktion, der Cosinus-Ähnlichkeit, die wir später benötigen:

```
(define (cosine-similarity vector1 vector2)
  (/
    (vec-mult vector1 vector2)
    (*
      (sqrt (vec-mult vector1 vector1))
      (sqrt (vec-mult vector2 vector2)))))
```

Diese Funktion verwendet die Funktion `vec-mult`, die das Skalarprodukt zweier Vektoren berechnen soll. Vektoren werden als Listen von Zahlen dargestellt. Zur Erinnerung: Das Skalarprodukt zweier Vektoren ist die Summe der Produkte der Komponenten der Vektoren. Also,
`(vec-mult '(1 1 0) '(1 0 1))` = `(+ (* 1 1) (* 1 0) (* 0 1))`.

1. (K) Schreiben Sie die Prozedur `vec-mult`, die zu zwei Vektoren (Listen von Zahlen) das Skalarprodukt berechnet.

   Hinweis: Multiplizieren Sie die jeweils ersten Komponenten der Vektoren und addieren Sie das Ergebnis zum Skalarprodukt der restlichen Vektorkomponenten. Das Skalarprodukt zweier leerer Vektoren ist 0. Sie dürfen voraussetzen, dass die Vektoren gleich viele Komponenten haben.

   - Berechnen Sie `(cosine-similarity '(1 0) '(0 1))`
   - Berechnen Sie `(cosine-similarity '(1 1) '(1 1))`

2. (K) Definieren Sie eine Struktur zum Speichern von Dokumenten. Ein Dokument hat einen Namen und einen Inhalt. Der Inhalt soll hier eine Liste von Symbolen sein (die Wörter des Dokuments, z.B. `'(Dies ist der Inhalt eines Dokuments)`). Definieren Sie zwei Dokumente:
   - `doc1` mit dem Namen `'doc1` und dem Inhalt `'(mouse keyboard screen)`.
   - `doc2` mit dem Namen `'doc2` und dem Inhalt `'(pascal java scheme)`.

3. Schreiben Sie die Funktion `index`, die einen Index zu einer Liste von Dokumenten erzeugt. Ein Index ist eine Liste aller im Inhalt der Dokumente vorkommenden Symbole ohne Duplikate. Verwenden Sie für Dokumente die Struktur aus Aufgabe 6.2. Hinweis: Eine leere Liste von Dokumenten ergibt einen leeren Index. Ist die Liste der Dokumente nicht leer, so kann der Inhalt des ersten Dokumentes mit der Prozedur `append` an den Index der restlichen Dokumente angefügt werden. Zur Erinnerung: Die Prozedur `append` verbindet mehrere Listen, z.B. `(append '(a b) '(c d))` = `'(a b c d)`. Verwenden Sie die Prozedur `remove-duplicates` aus der Präsenzübung um Duplikate aus dem Index zu entfernen.
   - Berechnen Sie `(index (list doc1 doc2))`.
   - Berechnen Sie `(index (list doc1 doc1 doc2))` und überprüfen Sie, dass keine Duplikate vorkommen.
   - Definieren Sie `myindex` als `(index (list doc1 doc2))`.

Der Index aus der vorigen Aufgebe wird verwendet, um Anfragen und Dokumente in Wortvektoren umzuwandeln. Ein Wortvektor zu einem Dokuement oder einer Anfrage enthält für jedes Element des Index eine 0 oder eine 1. Eine 0 bedeutet: Das entsprechende Wort aus dem Index kommt nicht im Dokumentinhalt / der Anfrage vor, eine 1 bedeutet es kommt vor. Dabei spielt es keine Rolle, dass im Dokumentinhalt / der Anfrage auch Symbole vorkommen, die nicht im Index sind.
Beispiel: Der Index sei `'(this list is an index)` und die Anfrage `'(this is a query)`, so ist der zugehörige Wortvektor `'(1 0 1 0 0)`. Dies bedeutet, dass `'this` und `'is` in der Anfrage vorkommen, alle anderen Symbole aus dem Index nicht.

4. Schreiben Sie eine Funktion `word-vector`, die eine Liste von Symbolen (die Anfrage) oder ein Dokument übergeben bekommt sowie eine zweite Liste von Symbolen (den Index) und den zugehörigen Wortvektor zurückliefert. Hinweis: Implementieren Sie die Prozedur zunächst nur für Anfragen, also Symbollisten, nicht für Dokumente. Verwenden Sie Rekursion. Bei einem

leeren Index ist der Wortvektor vollständig und es kann die leere Liste zurückgegeben werden. Bei nicht-leerem Index müssen Sie überprüfen ob das erste Wort im Index in der Symbolliste vorkommt und dementsprechend eine 1 oder 0 vor dem restlichen Wortvektor einfügen. Hierzu können Sie die Prozedur `cons` verwenden. Anschließend fügen Sie zu Beginn der Prozedur eine Behandlung heterogener Daten ein, die im Falle eines Dokumentes als erstem Parameter die Prozedur rekursiv mit dem Dokumenteninhalt aufruft.)

   - Berechnen Sie `(word-vector '(java mouse algol) myindex)`
   - Berechnen Sie `(word-vector doc1 myindex)`

Nun benötigen wir noch eine Struktur zur Ausgabe der Suchergebnisse. Diese soll einen Dokumentnamen zusammen mit einem Punktwert, dem `score` enthalten:

```
(define-struct result (name score))
```

5. Implementieren Sie eine Prozedur `compare`, die eine Suchanfrage (eine Liste von Symbolen), ein Dokument und einen Index erhält und daraus ein `result` erzeugt. Wenden Sie zur Berechnung des `score` die Funktion `cosine-similarity` auf den Wortvektor der Suchanfrage und den Wortvektor des Dokumenteninhaltes an. Für die Erstellung des Wortvektors verwenden Sie den übergebenen Index.

6. Schreiben Sie eine Prozedur `query`, die eine Suchanfrage und eine Liste von Dokumenten übergeben bekommt und eine Liste von Suchergebnissen zurückliefert. Hinweis: Verwenden Sie Rekursion. Sind keine Dokumente in der Liste, so ist das Ergebnis die leere Liste. Ansonsten verwenden Sie die Prozedur `compare`, um das Ergebnis des ersten Dokumentes zu berechnen und `cons` um die restlichen Ergebnisse anzuhängen.