Informatik I, Blatt Nr. 3, Abgabe: 13.11.2008 um 11 Uhr
http://proglang.informatik.uni-freiburg.de/teaching/info1/2008/

Hinweise zur Abgabe

Bitte reichen Sie Ihre Abgaben bis zum 13.11.2008 um 11 Uhr ein. Abgaben in elektronischer Form schicken Sie **per Email** an **Ihren** Tutor. Abgaben in Papierform werfen Sie bitte in den **Briefkasten** Ihrer Übungsgruppe im Geb. 051 im Erdgeschoss. Bei jeder Aufgabe ist angegeben, ob Sie elektronisch oder auf Papier abgegeben werden muss.

Bei allen Aufgaben, die Sie per Mail abgeben, müssen Sie sich an die Namenskonventionen der Aufgaben halten. Dies gilt sowohl für die Dateinamen der Abgabe, als auch für Namen von Funktionen. Bitte geben Sie bei der elektronischen Abgabe nur eine Zip-Datei ab. Diese muss alle in den Aufgaben angegebenen `.scm` Dateien (DrScheme) enthalten. Alle Dateien müssen sich in der Zip-Datei in einem Ordner befinden. Der Name dieses Ordners muss Ihrem Loginnamen für den Rechnerpool des Instituts für Informatik entsprechen. Geben Sie unter keinen Umständen Worddokumente usw. ab!

Achten Sie bei der Papierabgabe darauf, dass jedes Blatt Papier Ihrer Abgabe Ihren Namen, Ihre Übungsgruppe, die Blattnummer und den Namen Ihres Tutors trägt. Falls Ihre Papierabgabe aus mehreren Seiten besteht, tackern Sie die Blätter.

Sie können DrScheme im Pool verwenden (starten mit `drscheme`). Achten Sie darauf, dass Sie jeweils das richtige Sprachlevel ausgewählt haben!

Punktevergabe

Um für die Programmieraufgaben Punkte zu erhalten, folgen Sie den Konstruktionsanleitungen der Vorlesung: Schreiben Sie eine Kurzbeschreibung, führen Sie eine Datenanalyse durch und schreiben Sie einen Vertrag. Erstellen Sie dann die Testfälle und den Rumpf mit der passende Schablone. Vervollständigen Sie dann den Rumpf der Prozedur und vergewissern Sie sich, dass die Tests erfolgreich laufen. Schreiben Sie Hilfsprozeduren an den Stellen, an denen Sie Teilprobleme lösen!

Hinweis Endrekursion

Achtung: Wie in der Vorlesung gezeigt, verhindern Verträge die "korrekte" Ausführung endrekursiver Funktionen. Bitte kommentieren Sie in Aufgaben, in denen Sie eine endrekursive Funktion schreiben, den Vertrag aus.

1 Aufgabe

[Sprache: Die Macht der Abstraktion, 6 Punkte]

Benutzen Sie den in der Vorlesung definierten Datentyp für Listen um folgende Prozeduren zu schreiben:

a. Eine Prozedur `count-zeroes`, die die Anzahl von Nullen in einer Liste von Zahlen berechnet.

b. Eine Prozedur `contains>10?`, die feststellt, ob eine Liste von Zahlen eine Zahl enthält, die größer als 10 ist.

Benutzen Sie die Konstruktionsanleitung für Prozeduren über Listen!

Abgabe: elektronisch als `list-procedures.scm`.

2 Aufgabe

[Sprache: Die Macht der Abstraktion, 6 Punkte]

Schreiben Sie eine endrekursive Funktion `prod`, die eine Liste von Zahlen als Parameter erhält, und das Produkt der Zahlen als Ergebnis liefert. Die Funktion soll hierbei folgende Eigenschaft erfüllen:

> Falls eine der Zahlen in der Liste 0 ist, sollen alle später folgenden Zahlen nicht multipliziert und nicht angesehen werden.

Beispiel: Es soll für die Liste `(1 2 0 4 5 6 7 8 9 10)` das Produkt berechnet werden. Dann darf nur 1*2*0=0 berechnet werden, dannach muss abgebrochen werden. Die Zahlen 4, 5 usw. dürfen nicht angesehen werden.

Überprüfen Sie, ob Ihre Abgabe diese Eigenschaft erfüllt, indem sie mit dem Stepper von DrScheme prüfen, ob die Funktion auch wirklich nach dem Ansehen der ersten 0 direkt das Ergebnis liefert.

Abgabe: elektronisch als `prod.scm`.

3 Aufgabe

[Sprache: Die Macht der Abstraktion, 8 Punkte]

a. *(2 Punkte)* Schreiben Sie eine Funktion `fib`, die die Fibonacci-Zahl einer Zahl n berechnet. Verwenden Sie hierzu direkt die Definition der Fibonacci-Reihe:

$$\text{fib}(n) := \begin{cases} 0 & \text{falls } n = 0 \\ 1 & \text{falls } n = 1 \\ \text{fib}(n-1) + \text{fib}(n-2) & \text{sonst} \end{cases}$$

b. *(6 Punkte)* Verwenden Sie den Stepper von DrScheme, um zu beobachten, wie `(fib 4)` berechnet wird. Wie Sie sehen, wird der Wert für `(fib 2)` bei der Berechnung von `(fib 4)` zweimal berechnet. Schreiben Sie nun eine Funktion `fib-fast`. Diese Funktion soll ebenfalls die Fibonacci-Zahl berechnen, allerdings diese Mehrfachberechnungen nicht durchführen. Außerdem soll die Funktion `fib-fast` endrekursiv sein.

Abgabe: elektronisch als `fib.scm`.