Informatik I, Blatt Nr. 6, Abgabe: 04.12.2008 um 11 Uhr
http://proglang.informatik.uni-freiburg.de/teaching/info1/2008/

Hinweise zur Abgabe

Bitte reichen Sie Ihre Abgaben bis zum 04.12.2008 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 passenden 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 *[6 Punkte]*

Sei b ein Binärbaum. Definiere

- N(b) als die Anzahl der Knoten von b,
- L(b) als die Anzahl der Blätter von b,
- I(b) als die Anzahl der inneren Knoten von b und
- $E(b)$ als die Anzahl der in b vorkommenden leeren Bäume.

Beweisen Sie, dass für jeden Baum b gilt:

a. $N(b) + 1 = E(b)$

b. $N(b) - I(b) = L(b)$

Abgabe: Papier.

2 Aufgabe

[6 Punkte]

Die *Fibonacci*-Funktion auf den natürlichen Zahlen ist folgendermaßen definiert:

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

Beweisen Sie, dass $\mathit{fib}(n) = (\Phi^n - \Psi^n)/\sqrt{5}$
wobei $\Phi = (1 + \sqrt{5})/2$ und $\Psi = (1 - \sqrt{5})/2$.
Abgabe: Papier.

3 Aufgabe

[Sprache: Die Macht der Abstraktion, 8 Punkte]

Das folgende Zahlenmuster heißt das *Pascal'sche Dreieck*:

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
  ...
```

a. Die Zahlen an den seitlichen Kanten des Dreiecks sind allesamt Eins. Eine Zahl im Innern des Dreiecks ist die Summe der beiden Zahlen darüber. Schreiben Sie eine Prozedur `pascal`, die als Argumente die Nummer einer Zeile und die Nummer einer „Spalte“ innerhalb des Dreiecks akzeptiert, und die Zahl im Pascal'schen Dreieck berechnet. (Sowohl Zeilen- als auch Spaltennummer fangen bei Eins an.)

   Beispiel: `(pascal 5 3)` ergibt `6`

b. Schreiben Sie eine Prozedur, die als Argument die Nummer einer Zeile bekommt und die Zahlen der entsprechenden Zeile im Pascal'schen Dreieck in einer Liste zurückgibt.

   Beispiel: `(pascal-line 5)` ergibt `#<list 1 4 6 4 1>`

c. Schreiben Sie eine Prozedur, die als Argument die Nummer einer Zeile bekommt und bis zu dieser Zeile alle Zeilen des Pascal'schen Dreiecks in einer Liste zurückgibt.

   Beispiel: `(pascal-triangle 5)` ergibt

```
#<list #<list 1> #<list 1 1> #<list 1 2 1> #<list 1 3 3 1>
#<list 1 4 6 4 1>>
```

d. (Bonus-Aufgabe, 3 Punkte) Schreiben Sie eine Prozedur, die als Argument die Nummer einer Zeile bekommt und das Pascal'sche Dreieck bis zu dieser Zeile als eine schön formatierte Zeichenkette zurückgibt.
   Beispiel: `(pascal-triangle->string 6)` ergibt

```
"        1\n       1  1\n      1  2  1\n     1  3  3  1\n   1  4  6  4  1\n  1  5 10 10  5  1\n"
```

   Auf den ersten Blick ist die Schönheit der Zeichenkette nicht erkennbar. Wenn Sie die obige Zeichenkette jedoch an `write-string` Übergeben, druckt DrScheme folgendes Ergebnis aus:

```
        1
       1  1
      1  2  1
     1  3  3  1
   1  4  6  4  1
  1  5 10 10  5  1
```

Hinweise zur Bonusaufgabe:

- Folgende Funktionen können hilfreich sein: `number->string` und `string-append`. Schlagen Sie die Funktionen in der DrScheme-Hilfe nach, wenn Sie sie nicht kennen.
- Benutzen Sie `write-string` nicht in einer selbstdefinierten Prozedur. Ihre Prozeduren sollen eine Zeichenkette mit der Repräsentation des Pascal'schen Dreiecks zurückgeben. Der Aufruf von `write-string` sieht dann beispielsweise so aus: `(write-string (pascal-triangle->string 6))`.
- Einen Zeilenumbruch bewirken Sie mit `\n` in einer Zeichenkette.

Abgabe: Programm `pascal.scm`