Prof. Bernd Krieg-Brückner
Heino Gärtner
Shi Hui
Kolyang
Christoph Lüth

5. Übungsblatt

Ausgabe: 15. Januar 1998
Bearbeitungszeit: zwei Praktikumstermine

$\boxed{1}$ *Polynome* *Wichtung: 30%*

Definiert Funktionen `addP`, `mulP`, `negP` zum Addieren, Multiplizieren und Negieren von Polynomen, sowie eine Operation `evalP` zum Ausrechnen von Polynomen. Ein Polynom soll dabei als Liste der Koeffizienten dargestellt werden, so daß z. B. $3x^4 + x^2 - 8x + 10$ in der Listendarstellung als `poly1 = [3,0,1,-8,10]` geschrieben wird. Als Datentyp wird die Definition `type Poly = [Int]` verwendet. Im Test soll wenigstens das Resultat der Konstanten `test` angegeben werden.

```
poly2 = [2,-10,0,3]
poly3 = addP (negP poly1) (mulP poly2 poly1)
test = (poly3, map (evalP poly3) [-4,-3,0,1,2,20])
```

$\boxed{2}$ *Groß und genau* *Wichtung: 30%*

Eine Darstellung von positiven ganzen Zahlen beliebiger Genauigkeit sind Polynome, deren Koeffizienten alle im Bereich $0 \ldots (\texttt{base} - 1)$ liegen. Benutzt den Typ `type Big = Poly` und eine Definition der Basis `base` im Bereich $2 \ldots 100$. Der Wert einer Big-Zahl z ist dann `evalP z base`.

1. Schreibt eine Funktion `normal`, die Polynome in die beschriebene Normalformdarstellung konvertiert. Gebt Funktionen `toBig` und `fromBig` zur Umwandlung von und nach `Int` an.
2. Definiert Operationen `addB`, `subB`, `mulB` zum Addieren, Subtrahieren und Multiplizieren großer Zahlen.

$\boxed{3}$ *Permutationen* *Wichtung: 40%*

Eine Permutation einer Liste ist eine Liste, die die gleichen Elemente in beliebiger Reihenfolge enthält. Schreibt eine Funktion `perm :: Ord a => [a] -> [[a]]`, die sämtliche Permutationen einer Liste von Elementen, auf denen eine Ordnung definiert ist, berechnet. Z. B.:

```
perm [1,2,3]
==> [[1, 2, 3], [2, 1, 3], [2, 3, 1], [1, 3, 2], [3, 1, 2], [3, 2, 1]]
```

Eine Möglichkeit, dieses Problem zu lösen, besteht darin, sich zunächst eine Hilfsfunktion `insert_everywhere :: a-> [a]-> [[a]]` zu schreiben, die ein Element `a` an jeder Position der Liste `[a]` einfügt. Also z. B.

```
insert_everywhere 1 [2,3,4]
==> [[1, 2, 3, 4], [2, 1, 3, 4], [2, 3, 1, 4], [2, 3, 4, 1]]
insert_everywhere 3 [4]
==> [[3, 4], [4, 3]]
```

Die Funktion `perm` muß `insert_everywhere` dann zum schrittweisen Aufbau der Liste der Permutationen nutzen.