Übungsaufgaben zum SML-Kurs, SS 2003

Hans Leiß

Abgabe: Mi, 30.4.2003

1 Listenfunktionen

Aufgabe 1.1 Definiere mit Hilfe von `explode`, `implode` und Faltungen eine Funktion `spiegele : string -> string`, die Zeichenreihen spiegelt. (3 Punkte)

Aufgabe 1.2 Definiere den Datentyp `tree` durch

⟨*tree*⟩≡
```
datatype 'a tree = Leaf of 'a | Branch of 'a tree list;
```

von endlich verzweigenden (angeordneten) Bäumen mit Objekten vom Typ `'a` an den Blättern.

(i) Definiere die Spiegelung eines Baums, `mirror : 'a tree -> 'a tree`. Beispielsweise soll das

⟨*Original*⟩≡
```
val testbaum = Branch [Leaf 1, Branch [Leaf 2, Leaf 3]];
```

durch Anwenden von `speigele` in den Baum

⟨*Spiegelbild*⟩≡
```
Branch [Branch [Leaf 3, Leaf 2], Leaf 1]
```

umgewandelt werden. (3 Punkte)

(ii) Definiere mit einem Akkumulator eine Funktion

⟨*collect*⟩≡
```
collect : 'a tree -> 'a list
```

mit der die Blätter eines Baums (in der gleichen Anordnung wie im Baum) in einer Liste gesammelt werden. (5 Punkte)

Hinweis: definiere simultan mit `collect` eine Hilfsfunktion `collectL`, die die Blätter einer Liste von Bäumen sammelt.

(iii) Definiere mit Hilfe von Faltungen eine Funktion `sum : int tree -> int`, die die Summe der Blätter eines `int tree`'s bildet. (4 Punkte)