Institut für Informatik
der Universität München
Prof. Dr. M. Hofmann
Dr. J. Johannsen, Dr. M. Lange

WS 03/04

15.1.04

Übungen zur Vorlesung
Informatik I

Blatt 11

Abgabe der Hausaufgaben spätestens am 26.1.04, 11:00 Uhr. Programmieraufgaben über
`http://miles.tcs.informatik.uni-muenchen.de/info1/abgabe.php`,
schriftliche Aufgaben auf Papier im Briefkasten in der Theresienstraße 39, 1. Stock. Notieren Sie Namen, Matrikelnummern und Ihre Übungsgruppe auf den Blättern. Bearbeitung in Gruppen zu max. 3 Personen ist zulässig. Besprechung der Aufgaben in den Übungen ab 2.02.04.

Programmieraufgabe P-43 (`fold.ml`): **5 Punkte**

Der in der Vorlesung vorgestellte Datentyp α**vect** der *Reihungen* wird in Ocaml durch den Typ `'a array` realisiert. Für diesen gibt es die Operationen

- `Array.length : 'a array -> int` gibt die Länge eines Arrays an.
- `Array.get : 'a array -> int -> 'a` nimmt ein Array `a` und eine Zahl i, und liefert das Element von `a` an der Stelle i.
- `Array.sub : 'a array -> int -> int -> 'a array` nimmt ein Array `a` und zwei Zahlen n und m und liefert das Teilarray von `a`, welches an der Stelle n beginnt und die Länge m hat.

Das Array vom Typ `int array` der Länge 5, das die Zahlen von 0 bis 4 enthält, wird angegeben durch `[| 0 ; 1 ; 2 ; 3 ; 4 |]`.

Benutzen Sie die angegebenen Operationen auf Arrays, um eine Ocaml-Funktion `fold_balanced : ('a -> 'a -> 'a) -> 'a array -> 'a` zu schreiben, die eine zweistellige Funktion f und ein Array a nimmt und a mittels f faltet. Im Unterschied zu `fold_left` oder `fold_right` soll aber das Array jeweils so geteilt werden, dass die Längen der zwei Teile sich höchstens um 1 unterscheiden.

Beispiel: `fold_balanced f [|1;2;3;4;5|] = f (f 1 2) (f (f 3 4) 5).`

Programmieraufgabe P-44 (`schlange.ml`): **5 Punkte**

Eine *Schlange*, die Elemente vom Typ `'a` enthalten kann, läßt sich als Paar vom Typ `'a list * 'a list` folgendermassen realisieren. Sei (l_1, l_2) ein Paar von Listen. Beim Einfügen von x wird x vorne an l_2 angehängt. Beim Auslesen eines Elementes aus der Schlange wird das erste Element von l_1 zurückgeliefert. Ist l_1 leer, so wird zuerst l_2 nach l_1 Element für Element kopiert.

Implementieren Sie die folgenden Funktionen für Schlangen in Ocaml.

- `init : 'a list * 'a list` erzeugt eine leere Schlange.
- `nil : ('a list * 'a list) -> bool` prüft, ob eine Schlange leer ist.
- `enter : ('a list * 'a list) -> 'a -> ('a list * 'a list)` fügt ein Element in eine Schlange ein.

- `hd : ('a list * 'a list) -> 'a` liefert das erste Element einer Schlange.
- `tl : ('a list * 'a list) -> ('a list * 'a list)` liefert den Rest einer Schlange, d.h. die Schlange, die durch Entfernen des ersten Elementes entsteht.

Schriftliche Aufgabe S-45: **4 Punkte**
Zeigen Sie, dass ein binärer Baum der Höhe h höchstens $2^h - 1$ Elemente hat.

Programmieraufgabe P-46 (`baeume.ml`): **6 Punkte**
Erweitern Sie die in der Vorlesung vorgestellte Realisierung von binären Bäumen in Ocaml, indem Sie die folgenden Funktionen implementieren:

a) Die Funktion `subtree` vom Typ `'a bintree -> 'a bintree -> bool` testet, ob das erste Argument ein Teilbaum des zweiten Arguments ist.

b) Die Funktion `ispath` vom Typ `'a list -> 'a bintree -> bool` mit der folgenden Spezifikation: `ispath l t` soll true sein, wenn es im Baum `t` einen Pfad von der Wurzel zu einem Blatt gibt, dessen Knoten die Elemente in `l` sind.

c) Die Funktion `leaves` vom Typ `'a bintree -> 'a list` soll zu einem Baum die Liste seiner Blätter ausgeben.

   Zur Erinnerung: ein Blatt ist ein Teilbaum der Form `Build(x,Empty,Empty)`. Beispiel:

```
# leaves (Build(1,Build(2,Empty,Empty),Build(3,Empty,Empty)));;
- : int list = [2; 3]
```