Technische Universität München
Fakultät für Informatik

A. Berlea
berlea@in.tum.de
Raum: MI 02.07.053

Programmiersprachen

Wintersemester 2006/2007

4. Übungsblatt *23. November 2006*

Aufgabe 1:

Schreibe eine Funktion `approximate: float stream -> float -> float`, die den Grenzwert einer konvergenten Folge approximiert. `approximate s eps` liefert das erste Element im Strom s zurück, das die Präzision *eps* erfüllt.
Teste die Funktion, um die Wurzel einer positiven Zahl a zu berechnen. $\sqrt{a}$ kann durch ein Element der untenstehenden Folge approximiert werden:

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right) \text{ mit } n \geq 0 \text{ und } x_0 = 1$$

(Hinweis: Der Betrag einer reellen Zahl kann mit Hilfe der Funktion `Real.abs : real -> real` berechnet werden.)

Aufgabe 2:

Die Fibonacci Folge ist wie folgt definiert:

$$F_0 = 0$$
$$F_1 = 1$$
$$F_k = F_{k-1} + F_{k-2}\ \forall k \in \mathbb{N}, k \geq 2$$

a) Schreibe eine möglichst effiziente Funktion, die F_n für ein beliebiges $n \in \mathbb{N}$ berechnet!

b) Definiere einen Wert, der die (unendliche) Fibonacci-Folge repräsentiert!

Aufgabe 3:

a) Definiere einen Datentyp `('a * 'b) hash_map` zur Repräsentation von *Hash-Maps*. Eine *Hash-Map* ordnet einem Schlüssel vom Typ `'a` einen Wert vom Typ `'b` mit Hilfe einer *Hash-Tabelle* zu.

   Für den Datentyp `('a * 'b) hash_map` sollen folgende Funktionen definiert werden:

- Eine Funktion `init`, die eine leere *Hash-Map* erzeugt.
- Eine Funktion `put : ('a * 'b) hash_map -> 'a * 'b -> unit`, die als Argumente eine *Hash-Map* `hm` sowie ein als Zuordnung aufgefasstes Paar `(k,v)` erhält. Ein Aufruf dieser Funktion soll die Zuordnung `(k,v)` der *Hash-Map* hinzufügen. Ist dem Schlüssel `k` bereits ein Wert zugeordnet, so soll dieser überschrieben werden.
- Eine Funktion `list2hash_map`, die zu einer Liste vom Typ `('a * 'b) list` eine *Hash-Map* vom Typ `('a, 'b) hash_map` erzeugt.
- Eine Funktion `get : ('a * 'b) hash_map -> 'a -> 'b option`, die, falls dem übergebenen Schlüssel ein Wert `v` zugeordnet ist, `SOME v` zurückliefert. Andernfalls soll `NONE` zurückgeliefert werden.
- Eine Funktion `remove : ('a * 'b) hash_map -> 'a -> 'b option`, die, genau wie die Funktion `get`, den zugeordneten Wert zurückliefert und zusätzlich die Zuordnung aus der *Hash-Map* entfernt.
- Eine Funktion `iter : ('a * 'b -> unit) -> ('a * 'b) hash_map -> unit`, die als Argumente eine Funktion `f : ('a * 'b -> unit)` und eine *Hash-Map* `hm` erhält und dann die Funktion `f` auf alle Zuordnungen der *Hash-Map* anwendet.

b) Der Binomialkoeffizient $\binom{n}{k}$ läßt sich mit Hilfe der Formeln $\binom{n}{k} = \binom{n}{n-k}$ und $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ effizient rekursiv berechnen.

Definiere eine Funktion `bin : int * int -> int`, die den Binomialkoeffizienten berechnet. Verwende den in der vorherigen Aufgabe definierten Datentyp `hash_map`, um mehrfache Berechnungen des gleichen Binomialkoeffizienten zu vermeiden.