Kanaeva / Schweppe, FUB Informatik

# Übung zu Algorithmen und Programmieren III, WS 2001/2

Übung 3

Ausgabe: 30.10.01
Abgabe: 8.11.01 bis 14.00

**Aufgabe 1 (4 P)**
Spezifizieren Sie eine statische Methode (d.h. eine Prozedur): `static void combine(int[] a, int[] b)`, die jedes Element von `a` mit der Summe der Elemente von `b` multipliziert. Z.B. hat für `a = [2,5,3], b =[5,6]` beim Verlassen von `combine a` den Wert `[22,55,33]`. Implementieren Sie die Funktion in Haskell.

**Aufgabe 2 ( 4 P)**
a) Es gibt im allgemeinen verschiedene Invarianten I, die {I} while (B) S {Q} erfüllen. Sind I' und I'' zwei solcher Invarianten. Sind dann auch I' $\wedge$ I'' und I'$\vee$ I'' Invarianten? (Beweis)
b) Definieren Sie anhand einer Schlussregel die Semantik der '`repeat S until B`', die eine Anweisung so lange ausführt, bis die Bedingung `B` erfüllt ist.

**Aufgabe 3 ( 10 P)**
a) Für n > 0 berechnet das folgende Programm die größte Zweierpotenz, die kleiner gleich n ist. Geben Sie die Schleifeninvariante und die Terminierungsfunktion an.

{n > 0}
```
i = 1;
```
$\{n > 0 \wedge i == 1\}$
```
while  (2*i ≤ n) {
  i = 2*i
}
```
$\{0 < i \leq n < 2i \wedge \exists p: i == 2^p\}$

b) Beweisen Sie:

{x ≥ y ∧ y ≥ 0}
```
while  (x != y) {
   y++;
   x = x – x mod y
}
```
$\{x == y\}$

**Aufgabe 4 ( 6 P)**
Implementieren Sie bitte in Java die in der 2. Aufgabe des 2. Übungsblattes beschriebene Datenstruktur. Benutzen Sie dabei doppeltverkettete Listen und gehen Sie bei der Implementierung von der folgenden informellen Spezifikation aus:

```
public class PrioQueue {
// Die Schlange wird als doppeltverkettete Liste repräsentiert, in der die
// Einträge nach Priorität geordnet sind. Die Einträge der Schlange sind
// vom Typ T (int, String, char, ...).

final int MAXLength = 7;
Element head, tail;
```

```
public boolean insert (T elem, int prio);
// Vor.: Die Schlange ist nicht voll.
// Eff.: Die Schlange ist um elem (mit prio) erweitert.

public Element delete ();
// Vor.: Die Schlange ist nicht leer.
// Eff.: Die Schlange ist um das am längsten wartende Element
//       mit der höchsten Priorität verringert.

public PrioQueue flush();
// Vor.: -
// Eff.: Das Ergebnis ist eine leere Schlange.

class Element {
  T element;
  int priority;
  Element succ, pred;
}
}
```

Testen Sie Ihre `PrioSchlange` für einen geeigneten Typ `T`. Stellen Sie dabei sicher, dass der Überlauf getestet wird.