Technische Universität München
Fakultät für Informatik
Prof. Dr. H. Seidl

SS 2005
Übungsblatt 3
6. Mai 2005

Übungen zu Einführung in die Informatik II

Aufgabe 5 Asserts in Java

Die unten dargestellte Klasse `OrderedSet` implementiert geordnete Mengen als Liste. In der Realisierung der beiden Methoden `isIn` und `add` für eine Menge wird angenommen, dass die verwaltete Liste sortiert ist.

```
public class OrderedSet<E extends Comparable<E>>{
    public List<E> elements;

    public boolean isIn(E elem){
        List<E> l = elements;
        while(l!=null){
            if (l.info.compareTo(elem)<0) l = l.next;
            else if (l.info.compareTo(elem)==0) return true;
            else return false;
        }
        return false;}

    public void add(E elem){
        if (elements==null){
            elements = new List<E>(elem);
            return;
        }

        List<E> l = elements;
        while(l!=null){
            if (l.info.compareTo(elem)<0) {
                if (l.next==null) l.next = new List<E>(elem);
                else  l = l.next;
            }
            else if (l.info.compareTo(elem)==0) return;
            else l.insert(elem);
        }
    }
}
```

Die Klasse `List` ist wie folgt definiert:

```
public class List<E>{
    public E info;
    public List<E> next;

    public List(E e){
        info = e;
        next = null;
   }
}
```

Schreiben Sie eine Methode `public boolean sorted()`, die testet, ob die verwaltete Liste sortiert ist. Benutzen Sie diese Methode, um den Java-Code mit `assert` Aufrufen zu versehen, die

überprüfen, dass die Liste vor und nach jedem Methodenaufruf der Klasse `OrderedSet` sortiert ist.

Aufgabe 6 Verifikation: Berechnung des Quadrates einer Zahl

Gegeben sei folgendes Mini-Java-Programm aus der Vorlesung zur Berechnung des Quadrates einer Zahl a:

```
int a, i, x, z;
a = read();
i = 0;
x = -1;
z = 0;
while (i != a) {
        x = x + 2;
        z = z + x;
        i = i + 1;
}
assert(z==a*a);
write(z);
```

a) Erstellen Sie den Kontrollflussgraphen des Programms. Der Kontrollflussgraph soll dabei derart beschaffen sein, dass Zusicherungen an jedem Programmpunkt eingetragen werden können.

b) Wiederholen Sie die Vorgänge aus der Vorlesung zur Bestimmung einer geeigneten Zusicherung an jedem Programmpunkt und zur Überprüfung der lokalen Konsistenz der Zusicherungen.

c) Transformieren Sie das Programm so, dass es immer terminiert. Verifizieren Sie durch eine geeignete Erweiterung der Zusicherungen, dass das Programm terminiert.

Aufgabe 7 Verifikation: Summe der ersten n Zahlen

Die Summe der ersten n natürlichen Zahlen lässt sich mit Hilfe des folgenden Programms berechnen:

```
n = read ();
if (n < 0) {
     n = -n;
}
int i = 0;
int s = 0;
while (i < n){
     i = i + 1;
     s = s + i;
}
write (s);
```

a) Erstellen Sie den Kontrollflussgraphen des Programms. Der Kontrollflussgraph soll dabei derart beschaffen sein, dass Zusicherungen an jedem Programmpunkt eingetragen werden können.

b) Bestimmen Sie eine geeignete Zusicherung an jedem Programmpunkt. Gehen Sie dabei folgendermaßen vor: geben Sie eine Zusicherung an, die am Ende des Programms gelten soll; bestimmen Sie eine geeignete Schleifeninvariante; berechnen Sie die Zusicherungen an den übrigen Programmpunkten.

c) Verifizieren Sie die Konsistenz der angegebenen Zusicherungen.