

Übung zur Vorlesung Programmierung

Aufgabe T10

Gegeben sei folgendes Java-Programm P :

$\langle\varphi\rangle$ (Vorbedingung)

```
z = 0;
while (i < x){
  i = i + 1;
  j = 0;
  while (j < x){
    j = j + 1;
    z = z + x;
  }
}
```

$\langle\psi\rangle$ (Nachbedingung)

Als Vorbedingung φ für das oben aufgeführte Programm P gelte $x \geq 0$ und als Nachbedingung ψ gelte $z = x^3$. Beweisen Sie die Gültigkeit des Hoare-Tripels $\langle\varphi\rangle P \langle\psi\rangle$ unter Verwendung des Hoare-Kalküls (inklusive Terminierung) oder widerlegen Sie die Gültigkeit durch Angabe eines geeigneten Gegenbeispiels.

Aufgabe T11

Bestimmen Sie mit der Monte-Carlo-Methode das Volumen einer Kugel mit Radius 6 cm, aus der ein Zylinder mit Radius 3 cm und endloser Länge so ausgeschnitten wurde, daß die Mittelachse des Zylinders genau 3 cm am Mittelpunkt der Kugel vorbeigeht.

Aufgabe H10 (13 Punkte)

Gegeben sei folgendes Java-Programm P :

$\langle\varphi\rangle$ (Vorbedingung)

```
i = 0;
max = 0;
while (i < n){
  if (a[i] > max){
    max = a[i];
  }
  i = i + 1;
}
```

$\langle\psi\rangle$ (Nachbedingung)

Als Vorbedingung φ für das oben aufgeführte Programm P gelte $n \geq 0$ und als Nachbedingung ψ gelte $\max = \max(\bigcup_{k=0}^{n-1} \{a[k]\} \cup \{0\})$. Beweisen Sie die Gültigkeit des Hoare-Tripels $\langle\varphi\rangle P \langle\psi\rangle$ unter Verwendung des Hoare-Kalküls (inklusive Terminierung) oder widerlegen Sie die Gültigkeit durch Angabe eines geeigneten Gegenbeispiels.

Aufgabe H11 (3+5+3+2+2 Punkte)¹

Gegeben ist der Rumpf einer zirkulär doppelt-verketteten Liste. In so einer Liste kann von jedem Element auf das vorherige und das nachfolgende Element der Liste zugegriffen werden. Die Elemente formen einen Kreis, die Liste hat also kein Anfang oder Ende.

1. Implementieren Sie die Methode **public int** *size()*, welche die Anzahl an Elementen in der Liste zurückgibt.
2. Implementieren Sie die Methode **public** *CircularDoubleLinkedList insert(int value)*. Die Methode soll ein neues Element zur Liste hinzufügen und eine Referenz auf dieses Element zurückgeben.
3. Implementieren Sie die Methode **public boolean** *checkStructure()* um folgende Invariante der Liste zu prüfen: ein Listenelement hat ein Vorgängerelement *previous* und ein Nachfolgerelement *next*, beide dürfen nicht *null* sein. Weiterhin gilt für jedes Listenelement, daß der Vorgänger des Nachfolgers das Element selbst ist; genauso verhält es sich mit dem Nachfolger des Vorgängers. Die Methode *checkStructure()* soll diese Invariante für das Element, für welches sie aufgerufen wird, überprüfen und genau dann *true* zurückgeben, wenn diese Invariante gegeben ist.²

Rufen sie *checkStructure()* in einem **assert** vor und nach dem Ausführen jeder **public** Methode für jedes geänderte Element sowie jedes Element, daß eine Referenz auf ein geändertes Element besitzt, auf. Prüfen Sie mit einem solchen **assert** auch, ob die Invariante am Ende jedes Konstruktors für das gerade erzeugte Element gilt.

- (a) Stellt diese Invariante sicher, daß die Liste immer kreisförmig ist?
- (b) Stellt diese Invariante sicher, daß die Liste, welche von *insert* zurückgegeben wird, auch alle Elemente der ursprünglichen Liste beinhaltet?

```
class CircularDoubleLinkedList {
    public CircularDoubleLinkedList previous, next;
    public int value;
    public CircularDoubleLinkedList(int value){
        this.value = value;
        this.previous = this;
        this.next = this;
        assert(this.checkStructure());
    }
    private CircularDoubleLinkedList(int value,
        CircularDoubleLinkedList previous, CircularDoubleLinkedList next){
        this.value = value;
        this.previous = previous;
        this.next = next;
    }
    public CircularDoubleLinkedList insert(int value) { //TODO }
    public int size() { //TODO }
    public boolean checkStructure() { //TODO }
}
```

¹Bitte Quelltext für die Abgabe ausdrucken und zusätzlich per E-mail an den jeweiligen Tutor senden.

²Korrigiert: *checkStructure* wird nicht mehr im privaten Konstruktor aufgerufen.