

## Übungen zu Grundlagen der Softwarezuverlässigkeit

Abgabe bis zum

### Aufgabe 3.1      weakest precondition

Berechnen Sie für folgende Programmfragmente die weakest precondition mittels Kommentierung des Programmcodes. Bei Division durch 2 soll notfalls der Nachkommabereich abgeschnitten werden.

(a)  $\mathcal{C}_a \equiv$   
   $a := x;$   
   $b := y;$   
   $d := |a-b|;$   
   $m := a+b;$   
   $a := (m-d)/2;$   
   $b := (m+d)/2;$   
   $\{ a = \min\{x,y\}, b = \max\{x,y\} \}$

(b)  $\mathcal{C}_b \equiv$   
   $k := n;$   
   $f := 0;$   
   $g := 1;$   
  **while**  $k > 0$  **loop**  
     $t := g;$   
     $g := f + g;$   
     $f := t;$   
     $k := k - 1;$   
  **end;**  
   $\{k = 0 \wedge f = F_n \wedge g = F_{n+1}\}$

*Hinweis*

Mit  $F_0 := 0, F_1 = 1, F_2 = 1, F_3 = 2, \dots$  seien die Fibonacci-Zahlen bezeichnet.

(c)  $\mathcal{C}_c \equiv$   
   $tmp := b[i];$   
   $b[i] := b[k];$   
   $b[k] := tmp;$   
   $\{ b[i] \geq b[k] \}$

### Aufgabe 3.2      weakest precondition für repeat-until

Geben Sie entsprechend der Formulierung der Halteprädikate  $H_0, H_1, \dots$  für eine while-Schleife Halteprädikate für die repeat-until-Schleife an. Begründen Sie Ihre Konstruktion und berechnen Sie damit für folgendes Programmfragment die ersten drei Halteprädikate für die Bedingung  $P \equiv \{n = 1\}$ :

```
 $\mathcal{C} \equiv$   
  f := 1;  
  repeat  
    f := f * n;  
    n := n - 1;  
  until n < 2 end;
```

### Aufgabe 3.3      Generierung von Testfällen

Generieren Sie für das Programm

```
 $\mathcal{C}_b \equiv$   
  x := 1;  
  y := z;  
  b := 1;  
  while y > 0 loop  
    r := y / 2;  
    if 2 * r < y then  
      x = x * b;  
    end;  
    b := b * b;  
    y = r;  
  end;
```

einen Testfall, für welchen der Rumpf der while-Schleife genau zweimal durchlaufen wird, wobei *nur* im zweitem Durchlauf die if-Bedingung erfüllt sein soll. Gehen Sie entsprechend der Vorlesung vor, d.h. transformieren Sie das Programm  $\mathcal{C}$  durch Entrollen der Schleife und Beschneiden der Verzweigungen zunächst in ein Programm  $\mathcal{C}'$ , und berechnen Sie dann  $\text{wp}(\mathcal{C}', \mathbf{true})$ . Lesen Sie hieraus einen konkreten Testfall ab.

*Hinweise*

Nehmen Sie zur Vereinfachung an, dass alle Variablen nur nicht-negative Werte annehmen, wobei notfalls bei 0 abgeschnitten wird. Entsprechend wird bei Division durch 2 notfalls der Nachkommabereich abgeschnitten.

### Aufgabe 3.4      Kreise in balancierten Graphen

Entsprechend der Vorlesung heißt ein gerichteter Graph  $G = (V, E)$  balanciert, falls er stark zusammenhängend ist und für jeden Knoten die Anzahl der eingehenden Kanten gleich der Anzahl der ausgehenden Kanten ist. Dabei dürfen Kanten  $(u, w)$  auch mehrmals in  $E$  auftreten, d.h.  $E$  ist eine Multimenge (aka. Liste).

Zeigen Sie, dass folgender Algorithmus immer einen Kreis  $\zeta = \zeta_0 \dots \zeta_n$  (mit  $\zeta_0 = \zeta_n$ ) in  $G$  konstruiert.

```
i := 0;
E0 := E;
Wähle  $\zeta_0 \in V$  beliebig;
while  $\zeta_i E_i \neq \emptyset$  loop
    Wähle  $\zeta_{i+1} \in \zeta_i E_i$  beliebig;
     $E_{i+1} := E_i -_{\mathcal{M}} \{(\zeta_i, \zeta_{i+1})\}_{\mathcal{M}}$ ;
     $i := i + 1$ ;
end;
```

Dabei sei  $uE := \{w \in V \mid (u, w) \in E\}$  die Menge der Nachfolger von  $u \in V$ . 'M' kennzeichnet hierbei die Operationen bezüglich Multimengen.

### Allgemeine Hinweise

- Vorlesungswebseite:  
<http://www.fmi.uni-stuttgart.de/szs/teaching/ws0405/grundsoftzuv/>
- Fragen zu den Übungen oder dem Stoff der Vorlesung können Sie im *Info-Forum* im Bereich *Sichere und Zuverlässige Software* posten:  
<http://swt.uni-stuttgart.de/forum>
- Für sonstige Fragen wenden Sie sich bitte an:  
Michael.Luttenberger@fmi.uni-stuttgart.de, Raum 1.342.