GRUNDLAGEN DER THEORETISCHEN INFORMATIK SS 2011

Prof. Dr. Sebastian Iwanowski

Übungsblatt 05 (5 Aufgaben) S.1/4

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Aufgabe 1)

Betrachten Sie folgendes Programm:

```
{ n,f,k ∈ N }          φ

f := 1;
k := n;

while(k > 0) do
  begin
     k := k - 1;
     f := k • f;
  end

{ Nachbedingung }      ψ
```

a) Formulieren Sie die Invariantenbedingungen, die für jeden Schleifendurchlauf gültig sind. Beweisen Sie das durch vollständige Induktion.

b) Zeigen Sie, dass dann die Schleife irgendwann terminiert (wann genau?). Formulieren und beweisen Sie direkt unter Verwendung von a), was diese dann berechnet hat.

c) Verändern Sie die Abbruchbedingung der Schleife so, dass das Programm eine Fakultät berechnet. Welche Fakultät berechnet das Programm genau? Beweisen Sie auch das unter Verwendung von a)

Aufgabe 2)

Betrachten Sie folgendes Programm:

```
Gegeben seien n Zahlen a[1] ... a[n] ∈ ℚ.

m := 0;
k := 0;
while (k < n) do
  begin
    m := k * m;
    k := k + 1;
    m := (m + a[k])/k ;
  end
```

a) Geben Sie die Invariantenbedingungen m_i und k_i an, die nach jedem Schleifendurchlauf erfüllt sind! Brauchen Sie Vorbedingungen dafür?
b) Beweisen Sie die Gültigkeit der Invariantenbedingungen von a) mit vollständiger Induktion!
c) Geben Sie an, nach wie vielen Durchläufen die Schleife abbricht und folgern Sie mit Hilfe von a) eine Nachbedingung für m (falls Sie in a) eine Vorbedingung formuliert haben, dürfen Sie diese weiterhin benutzen)!

Aufgabe 3)

Zeigen Sie, dass das folgende Programm DIV und MOD ausrechnet, indem Sie eine geeignete Invariantenbedingung für die Schleife formulieren und mit vollständiger Induktion beweisen und daraus die am Ende angegebene Nachbedingung folgern:

```
{ (x ≥ 0) ∧ (y > 0) ∧ x,y ∈ ℤ } φ
div := 0;
mod := x;
while mod ≥ y do
   begin
      mod := mod - y;
      div := div + 1;
   end
{ (x = div * y + mod) ∧ (0 ≤ mod < y) } ψ
```

Zusatz für Programmierfreaks: Formulieren Sie das entsprechende Programm so um, dass es auch für negative Zahlen DIV und MOD richtig berechnet (also nicht so wie Pascal!).

Aufgabe 4)

Das folgende Programm berechnet eine natürliche Potenz von b deutlich schneller als durch die einfachere Methode, n mal b mit sich selbst zu multiplizieren. Hier soll nicht bewiesen werden, dass das schneller geht, sondern nur dass das Programm überhaupt das gewünschte Ergebnis liefert.

Die Geschwindigkeitsgewinn kann für große n leicht durch ein eigenes Programm nachgeprüft werden. Hinweis: Arbeiten Sie in herkömmlichen Sprachen wie Pascal nicht mit Integerwerten, weil Sie sonst MAXINT überschreiten.

```
{ b ∈ Z, e ∈ N }                                        φ

   result := 1;
   base := b;
   exp := e;

   while exp > 0
     begin
       if (exp MOD 2 = 1) then
         begin
           result := result * base;
           exp := exp – 1;
         end
       else
         begin
           base := base * base;
           exp := exp DIV 2;
         end;
     end; {while}

{ result = b^e }                                        ψ
```

a) Beweisen Sie mit vollständiger Induktion über k: $b^e = base_k^{\exp_k} \cdot result_k$
(Hierbei sind $base_k$, exp_k und $result_k$ die Werte der Variablen nach dem k-ten Schleifendurchlauf)
Tipp: Unterscheiden Sie in Ihrem Induktionsbeweis die beiden Fälle, ob exp_k gerade ist oder nicht!

b) Folgern Sie daraus, dass das Programm mit der gewünschten Nachbedingung stoppt.

Aufgabe 5) (Klausuraufgabe SS 2010)

Gegeben sei der folgende Programmausschnitt:

```
1  {n > 0 vom Typ integer}
2  k := 0; s := 1;
3  while (k < n) do
4  begin
5     k := s - 1;
6     s := s + 1;
7  end
```

a) Formulieren Sie Bedingungen für k und s, die nach dem i-ten Schleifendurchlauf erfüllt sind.

b) Beweisen Sie die Bedingungen aus a) mit vollständiger Induktion über i.

c) Geben Sie Nachbedingungen für k und s an.

d) Begründen Sie die Aussage von c): Hierfür dürfen Sie alles bisher gezeigte verwenden.