Aufgabe 1)

Betrachten Sie folgendes Programm:

```
{ n,s,k ∈ N }        φ

s := 1;
k := n;

while(k > 0) do
  begin
     k := k - 1;
     s := k + s;
  end

{ Nachbedingung }    ψ
```

a) Formulieren Sie die Belegungswerte, 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 das Programm so, dass es die Summe von 1 bis n berechnet. Sie dürfen dafür die Initialisierungswerte oder die Abbruchbedingung ändern, nicht aber die Reihenfolge der Anweisungen.

Aufgabe 2)

Gegeben sei folgendes Programm:

```
{n: integer}

k := 0; s := 0;
while k < n do
begin
  s := k + s;
  k := k + 1;
end {while}
```

a) Was berechnet dieses Programm? Geben Sie die genaue Abhängigkeit von n an!
b) Beweisen Sie a)!
   <u>Hinweis:</u> Sie sollten mit diesem Teil beginnen, also erst mal durch Ausprobieren die Invariantenbedingungen für s und k bestimmen, nachsehen, wann das Programm abbricht, und dann erst die genaue Abhängigkeit des Ergebnisses von n angeben.

Aufgabe 3)

Gegeben sei folgendes Programm:

```
{n: integer}

k := 1; s := 0;
while k < n do
begin
  s := k + s;
  k := k + s;
end {while}
```

a) Zeigen Sie, dass für alle i gilt: $s_i = F_{2i}$ und $k_i = F_{2i+1}$.[1]
b) Spezifizieren Sie genau (in Worten), was am Ende ausgerechnet wird.

Aufgabe 4)

Betrachten Sie folgendes Programm:

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

k := 1;
while (k < n) do
  begin
      k := k + 1;
      d := a[k] – a[k-1];
      if m > d
        then
            m := d;
  end
```

a) Geben Sie eine Nachbedingung für m an! Brauchen Sie Vorbedingungen dafür?

b) Bestimmen Sie die Invariantenbedingungen m_i, k_i und d_i, die nach jedem Schleifendurchlauf erfüllt sind und beweisen Sie das mit vollständiger Induktion!

c) Beweisen Sie, dass die Schleife terminiert und folgern Sie dann die in a) angegebene Nachbedingung!

d) Ändern Sie das Programm so ab, dass es den Betrag des kleinsten Abstands ausgibt, der zwischen zwei hintereinander folgenden Zahlen auftreten kann!

[1] F_i ist die i-te Fibonaccizahl: $F_0 = 0$ $F_1 = 1$, $F_2=1$, …

Aufgabe 5)

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
```

Tipp: Die Invariantenbedingung ist ein Teil der Nachbedingung. Vergleichen Sie diese Aufgabe auch mit Aufgabe 2 des 4. Übungsblatts.

Aufgabe 6)

Das folgende Programm berechnet eine natürliche Potenz von b deutlich schneller als 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.