

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
```

- Formulieren Sie die Belegungswerte, die für jeden Schleifendurchlauf gültig sind. Beweisen Sie das durch vollständige Induktion.
- Zeigen Sie, dass dann die Schleife irgendwann terminiert (wann genau?). Formulieren und beweisen Sie direkt unter Verwendung von a), was diese dann berechnet hat.
- 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}
```

- Was berechnet dieses Programm? Geben Sie die genaue Abhängigkeit von n an!
- Beweisen Sie a)!

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}
```

- Zeigen Sie, dass für alle i gilt: $s_i = F_{2i}$ und $k_i = F_{2i+1}$.¹
- Geben Sie an, was am Ende ausgerechnet wird.

Aufgabe 4)

Betrachten Sie folgendes Programm:

Gegeben seien n Zahlen $a[1] \dots a[n] \in \mathbb{Q}$.

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

- Geben Sie eine Nachbedingung für m an! Brauchen Sie Vorbedingungen dafür?
- Bestimmen Sie die Invariantenbedingungen m_i , k_i und d_i , die nach jedem Schleifendurchlauf erfüllt sind und beweisen Sie das mit vollständiger Induktion!
- Beweisen Sie, dass die Schleife terminiert und folgern Sie dann die in a) angegebene Nachbedingung!
- 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)

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.

 $\{ b \in \mathbb{Z}, e \in \mathbb{N} \}$
 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}

```

 $\{ \text{result} = b^e \}$
 ψ

- a) Beweisen Sie mit vollständiger Induktion über k : $b^e = \text{base}_k^{\text{exp}_k} \cdot \text{result}_k$
 (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.