Prof. Dr. J. Giesl
D. Cloerkes, S. Dollase, N. Lommen, D. Meier, F. Meyer

Aufgabe 3 (Programmierung): (15 Punkte)

Implementieren Sie ein Programm, welches zu einem gegebenen Startdatum und einer gegebenen Anzahl `t` an Tagen ein Enddatum berechnet, sodass das Enddatum genau `t` Tage nach dem Startdatum liegt. Beispielsweise liegt der 05.11.2021 genau einen Tag nach dem 04.11.2021. Die Berechnung soll mithilfe *einer* ***einzigen*** *geeigneten Schleife* durchgeführt werden. Das Programm soll zudem weder `break` noch `continue` benutzen.

Das Programm fragt zunächst nach dem Startdatum. Hierzu wird der*die Benutzer*in nacheinander aufgefordert, den entsprechenden Tag des Monats, die Nummer des Monats und das Jahr einzugeben. Anschließend fragt das Programm nach der Anzahl `t` an Tagen. Alle Werte sollen als `int` eingelesen werden. In allen Berechnungen kann die Existenz von Schaltjahren vernachlässigt werden. Falls das eingegebene Startdatum nicht existiert oder `t` keine positive Zahl ist, so darf sich das Programm beliebig verhalten.

Ein Beispiellauf des Programms könnte also so aussehen:

```
Bitte geben Sie die Tageskomponente des Startdatums ein.
27
Bitte geben Sie die Monatskomponente des Startdatums ein.
2
Bitte geben Sie die Jahreskomponente des Startdatums ein.
2020
Bitte geben Sie die Anzahl an Tagen ein:
365
Das Datum 27.2.2021 befindet sich 365 Tage nach dem Startdatum.
```

Hinweise:

- Für nicht-negative ganze Zahlen berechnet der `Java`-Operator `%` die modulo-Funktion. D.h. für natürliche Zahlen n und d ist `n%d` gerade der Rest der Divison von n und d.
- Verwenden Sie die Klasse `SimpleIO` zum Einlesen und Ausgeben von Werten. Legen Sie die bereitgestellte Datei `SimpleIO.java` einfach im gleichen Verzeichnis wie ihre Lösung ab. Dann findet `Java` diese automatisch.

Lösung:

```java
public class Datumsrechner {
  public static void main(String args[]) {
    // Wir zaehlen von 0..30
    int tag =
      SimpleIO.getInt("Bitte geben Sie die Tageskomponente des Startdatums ein.") - 1;
    // Wir zaehlen von 0..11
    int monat =
      SimpleIO.getInt("Bitte geben Sie die Monatskomponente des Startdatums ein.") - 1;
    int jahr =
      SimpleIO.getInt("Bitte geben Sie die Jahreskomponente des Startdatums ein.");

    int anzahlTage = SimpleIO.getInt("Bitte geben Sie die Anzahl an Tagen ein.");

    for (int i = 0; i<anzahlTage; ++i) {
      switch(monat) {
        // Monat mit 28 Tagen (Februar)
        case 1 -> {
          if (tag == 27)
```

```
                ++monat;
              tag = (tag + 1) % 28;
            }

            // Monate mit 30 Tagen
            case 3,5,8,10 -> {
              if (tag == 29)
                ++monat;
              tag = (tag + 1) % 30;
            }

            // Alle anderen Monate haben 31 Tage
            default -> {
              if (tag == 30) {
                if (monat == 11)
                  ++jahr;
                monat = (monat + 1) % 12;
              }
              tag = (tag+1) % 31;
            }
          }
        }
        ++tag; ++monat;

        SimpleIO.output("Das Datum "+tag+"."+monat+"."+jahr+" befindet sich "
                        +anzahlTage+" Tage nach dem Startdatum.");
    }
}
```

Aufgabe 5 (Verifikation): (13 Punkte)

Gegeben sei folgendes Java-Programm über den Integer-Variablen `x`, `res` und `s`:

$\langle x > 0 \rangle$ (Vorbedingung)

```
  res = 1;
  s = 1;
  while (s < x) {
    res = res + 1;
    s = s + 2 * res - 1;
  }
```

$\langle \texttt{res} = \lceil \sqrt{x} \rceil \rangle$ (Nachbedingung)

Vervollständigen Sie die folgende Verifikation der partiellen Korrektheit des Algorithmus im Hoare-Kalkül, indem Sie die unterstrichenen Teile ergänzen. Hierbei dürfen zwei Zusicherungen nur dann direkt untereinander stehen, wenn die untere aus der oberen folgt. Hinter einer Programmanweisung darf nur eine Zusicherung stehen, wenn dies aus einer Regel des Hoare-Kalküls folgt.

Hinweise:

- Gehen Sie davon aus, dass keine Integer-Überlaufe stattfinden, d.h., behandeln Sie Integers als die unendliche Menge $\mathbb{Z}$.
- Sie dürfen beliebig viele Zusicherungs-Zeilen ergänzen oder streichen. In der Musterlösung werden allerdings genau die angegebenen Zusicherungen benutzt.

- Bedenken Sie, dass die Regeln des Kalküls syntaktisch sind, weshalb Sie semantische Änderungen (beispielsweise von $x + 1 = y + 1$ zu $x = y$) nur unter Zuhilfenahme der Konsequenzregeln vornehmen dürfen.
- Es empfiehlt sich oft, bei der Erstellung der Zusicherungen in der Schleife von unten (d. h. von der Nachbedingung aus) vorzugehen.
- $\lceil x \rceil$ ist die kleinste Zahl $n \in \mathbb{Z}$, sodass $n \geq x$ gilt. Insbesondere gilt also $n = \lceil \sqrt{x}\, \rceil$ genau dann, wenn $(n-1)^2 < x \leq n^2$ gilt.

Lösung:

```
                              ⟨x > 0⟩
                              ⟨x > 0 ∧ 1 = 1 ∧ 1 = 1⟩
res = 1;
                              ⟨x > 0 ∧ res = 1 ∧ 1 = 1⟩
s = 1;
                              ⟨x > 0 ∧ res = 1 ∧ s = 1⟩
                              ⟨s = res² ∧ (res − 1)² < x⟩
while (s < x) {
                              ⟨s = res² ∧ (res − 1)² < x ∧ s < x⟩
                              ⟨s + 2 · (res + 1) − 1 = (res + 1)² ∧ ((res + 1) − 1)² < x⟩
   res = res + 1;
                              ⟨s + 2 · res − 1 = res² ∧ (res − 1)² < x⟩
   s = s + 2 * res - 1;
                              ⟨s = res² ∧ (res − 1)² < x⟩
}
                              ⟨s = res² ∧ (res − 1)² < x ∧ ¬(s < x)⟩
                              ⟨res = ⌈√x⌉⟩
```

Um von der sechsten auf die siebte Zusicherung zu kommen, benutzen wir die Konsequenzregel. Aus $\texttt{s} = \texttt{res}^2$ folgt

$$(\texttt{res}+1)^2 = \texttt{res}^2 + 2\cdot\texttt{res} + 1 = \texttt{s} + 2\cdot\texttt{res} + 1$$

und mit $\texttt{s} < \texttt{x}$ gilt weiterhin $\texttt{res}^2 = ((\texttt{res}+1)-1)^2 < \texttt{x}$.
Um von der vorletzten auf die letzte Zusicherung zu kommen, benutzen wir ebenfalls die Konsequenzregel. Aus $(\texttt{res}-1)^2 < \texttt{x}$ folgt $\texttt{res} - 1 < \lceil\sqrt{\texttt{x}}\,\rceil$. Da jedoch $\texttt{s} = \texttt{res}^2 \geq \texttt{x}$ gilt, folgern wir $\texttt{res} \geq \lceil\sqrt{\texttt{x}}\,\rceil$ und damit $\texttt{res} = \lceil\sqrt{\texttt{x}}\,\rceil$.

Aufgabe 7 (Verifikation): (17 + 5 = 22 Punkte)

Gegeben sei folgendes Java-Programm über den Integer-Variablen a, b, x, res und y:

```
⟨b ≥ 0⟩                 (Vorbedingung)
   x = b;
   res = a;
   y = 1;
   while (x > 0) {
      if (x % 2 == 0) {
         y = 2 * y;
         x = x / 2;
      } else {
         res = res + y;
         y = 2 * y;
         x = (x - 1) / 2;
      }
   }
⟨res = a + b⟩                 (Nachbedingung)
```

a) Vervollständigen Sie die folgende Verifikation des Algorithmus im Hoare-Kalkül, indem Sie die unterstrichenen Teile ergänzen. Hierbei dürfen zwei Zusicherungen nur dann direkt untereinander stehen, wenn die untere aus der oberen folgt. Hinter einer Programmanweisung darf nur eine Zusicherung stehen, wenn dies aus einer Regel des Hoare-Kalküls folgt.

Hinweise:

- Gehen Sie davon aus, dass keine Integer-Überlaufe stattfinden, d.h., behandeln Sie Integers als die unendliche Menge $\mathbb{Z}$.
- Die Programmanweisung `x / 2` berechnet die Integer-Division x div 2.
- Sie dürfen beliebig viele Zusicherungs-Zeilen ergänzen oder streichen. In der Musterlösung werden allerdings genau die angegebenen Zusicherungen benutzt.
- Bedenken Sie, dass die Regeln des Kalküls syntaktisch sind, weshalb Sie semantische Änderungen (beispielsweise von $x+1 = y+1$ zu $x = y$) nur unter Zuhilfenahme der Konsequenzregeln vornehmen dürfen.
- Es empfiehlt sich oft, bei der Erstellung der Zusicherungen in der Schleife von unten (d. h. von der Nachbedingung aus) vorzugehen.

$\langle \texttt{res} = \texttt{a} + \texttt{b} \rangle$

b) Untersuchen Sie den Algorithmus P auf seine Terminierung. Für einen Beweis der Terminierung muss eine Variante angegeben werden und mit Hilfe des Hoare-Kalküls die Terminierung bewiesen werden.

Lösung:

a)

$\langle \texttt{b} \geq 0 \rangle$
$\langle \texttt{b} \geq 0 \wedge \texttt{b} = \texttt{b} \wedge \texttt{a} = \texttt{a} \wedge 1 = 1 \rangle$

```
x = b;
```

$\langle \texttt{x} \geq 0 \wedge \texttt{x} = \texttt{b} \wedge \texttt{a} = \texttt{a} \wedge 1 = 1 \rangle$

```
res = a;
```

$\langle \texttt{x} \geq 0 \wedge \texttt{x} = \texttt{b} \wedge \texttt{res} = \texttt{a} \wedge 1 = 1 \rangle$

```
y = 1;
```

$\langle \texttt{x} \geq 0 \wedge \texttt{x} = \texttt{b} \wedge \texttt{res} = \texttt{a} \wedge \texttt{y} = 1 \rangle$
$\langle \texttt{a} + \texttt{b} = \texttt{res} + \texttt{x} \cdot \texttt{y} \wedge \texttt{x} \geq 0 \rangle$

```
while (x > 0) {
```

$\langle \texttt{a} + \texttt{b} = \texttt{res} + \texttt{x} \cdot \texttt{y} \wedge \texttt{x} \geq 0 \wedge \texttt{x} > 0 \rangle$

```
  if (x % 2 == 0) {
```

$\langle \texttt{a} + \texttt{b} = \texttt{res} + \texttt{x} \cdot \texttt{y} \wedge \texttt{x} \geq 0 \wedge \texttt{x} > 0 \wedge \texttt{x}\%2 = 0 \rangle$
$\langle \texttt{a} + \texttt{b} = \texttt{res} + (\texttt{x} \text{ div } 2) \cdot 2 \cdot \texttt{y} \wedge \texttt{x} \text{ div } 2 \geq 0 \rangle$

```
    y = 2 * y;
```

$\langle \texttt{a} + \texttt{b} = \texttt{res} + (\texttt{x} \text{ div } 2) \cdot \texttt{y} \wedge \texttt{x} \text{ div } 2 \geq 0 \rangle$

```
    x = x / 2;
```

$\langle \texttt{a} + \texttt{b} = \texttt{res} + \texttt{x} \cdot \texttt{y} \wedge \texttt{x} \geq 0 \rangle$

```
  } else {
```

$\langle \texttt{a} + \texttt{b} = \texttt{res} + \texttt{x} \cdot \texttt{y} \wedge \texttt{x} \geq 0 \wedge \texttt{x} > 0 \wedge \neg(\texttt{x}\%2 = 0) \rangle$
$\langle \texttt{a} + \texttt{b} = \texttt{res} + \texttt{y} + ((\texttt{x} - 1) \text{ div } 2) \cdot 2 \cdot \texttt{y} \wedge (\texttt{x} - 1) \text{ div } 2 \geq 0 \rangle$

```
    res = res + y;
```

$\langle \texttt{a} + \texttt{b} = \texttt{res} + ((\texttt{x} - 1) \text{ div } 2) \cdot 2 \cdot \texttt{y} \wedge (\texttt{x} - 1) \text{ div } 2 \geq 0 \rangle$

```
    y = 2 * y;
```

$\langle \texttt{a} + \texttt{b} = \texttt{res} + ((\texttt{x} - 1) \text{ div } 2) \cdot \texttt{y} \wedge (\texttt{x} - 1) \text{ div } 2 \geq 0 \rangle$

```
    x = (x - 1) / 2;
```

$\langle \texttt{a} + \texttt{b} = \texttt{res} + \texttt{x} \cdot \texttt{y} \wedge \texttt{x} \geq 0 \rangle$

```
  }
```

$\langle \texttt{a} + \texttt{b} = \texttt{res} + \texttt{x} \cdot \texttt{y} \wedge \texttt{x} \geq 0 \rangle$

```
}
```

$\langle \texttt{a} + \texttt{b} = \texttt{res} + \texttt{x} \cdot \texttt{y} \wedge \texttt{x} \geq 0 \wedge \neg \texttt{x} > 0 \rangle$
$\langle \texttt{res} = \texttt{a} + \texttt{b} \rangle$

b) Wir wählen als Variante $V = \texttt{x}$. Hiermit lässt sich die Terminierung des Programms beweisen, denn für die einzige Schleife im Programm (mit Schleifenbedingung $B = \texttt{x > 0}$) gilt:

- $B \Rightarrow V \geq 0$, denn $B = \texttt{x} > 0 \Rightarrow \texttt{x} \geq 0$ und
- die folgende Ableitung ist korrekt:

$\langle \texttt{x} = m \wedge \texttt{x} > 0 \rangle$

```
if (x % 2 == 0) {
```

$\langle \texttt{x} = m \wedge \texttt{x} > 0 \wedge \texttt{x}\%2 = 0 \rangle$
$\langle \texttt{x} \text{ div } 2 < m \rangle$

```
  y = 2 * y;
```

$\langle \texttt{x} \text{ div } 2 < m \rangle$

```
  x = x / 2;
```

$\langle \texttt{x} < m \rangle$

```
} else {
```
$\langle \texttt{x} = m \wedge \texttt{x} > 0 \wedge \neg(\texttt{x}\%2 = 0) \rangle$
$\langle (\texttt{x} - 1) \text{ div } 2 < m \rangle$
```
  res = res + y;
```
$\langle (\texttt{x} - 1) \text{ div } 2 < m \rangle$
```
  y = 2 * y;
```
$\langle (\texttt{x} - 1) \text{ div } 2 < m \rangle$
```
  x = (x - 1) / 2;
```
$\langle \texttt{x} < m \rangle$
```
}
```
$\langle \texttt{x} < m \rangle$

Aufgabe 8 (Deck 2): (Codescape)

Lösen Sie die Missionen von Deck 2 des Codescape Spiels. Ihre Lösung für die Codescape Missionen wird nur dann für die Zulassung gezählt, wenn sie Ihre Lösung vor der einheitlichen Codescape Deadline am Samstag, den 22.01.2022, um 23:59 Uhr abschicken.

Lösung: