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

Allgemeine Hinweise:

- Die **Deadline** zur **Abgabe** der Hausaufgaben ist am **Donnerstag, den 04.11.2021, um 18:00 Uhr**.
- Der **Workflow** sieht wie folgt aus. Die Abgabe der Hausaufgaben erfolgt **im Moodle-Lernraum** und kann nur in **Zweiergruppen** stattfinden. Dabei müssen die Abgabepartner*innen **dasselbe Tutorium** besuchen. Nutzen Sie ggf. das entsprechende **Forum** im Moodle-Lernraum, um eine*n Abgabepartner*in zu finden. Es darf **nur ein*e** Abgabepartner*in die Abgabe hochladen. Diese*r muss sowohl die **Lösung** als auch den **Quellcode** der Programmieraufgaben hochladen. Die Bepunktung wird dann von uns für **beide** Abgabepartner*innen **separat** im Lernraum eingetragen. Die Feedbackdatei ist jedoch nur dort sichtbar, wo die Abgabe hochgeladen wurde und muss innerhalb des Abgabepaars **weitergeleitet** werden.
- Die **Lösung** muss als PDF-Datei hochgeladen werden. Damit die Punkte beiden Abgabepartner*innen zugeordnet werden können, müssen **oben** auf der **ersten Seite** Ihrer Lösung die **Namen**, die **Matrikelnummern** sowie die **Nummer des Tutoriums** von **beiden** Abgabepartner*innen angegeben sein.
- Der **Quellcode** der Programmieraufgaben muss als `.zip`-Datei hochgeladen werden und **zusätzlich** in der PDF-Datei mit Ihrer Lösung enthalten sein, sodass unsere Hiwis ihn mit Feedback versehen können.

  Auf diesem Blatt muss Ihre Codeabgabe Ihren vollständigen Java-Code in Form von `.java`-Dateien enthalten. Aus dem Lernraum heruntergeladene Klassen, etwa die Datei `SimpleIO.java`, dürfen nicht mit abgegeben werden.

  Stellen Sie sicher, dass Ihr Programm von `javac` **akzeptiert** wird, wenn die entsprechenden Klassen aus dem Lernraum hinzugefügt werden. Ansonsten werden keine Punkte vergeben.
- Einige Hausaufgaben müssen im Spiel Codescape gelöst werden. Klicken Sie dazu im Lernraum rechts im Block "Codescape" auf den angegebenen Link. Diese Aufgaben werden getrennt von den anderen Hausaufgaben gewertet.

Tutoraufgabe 1 (Überblickswissen):

a) Wie unterscheidet sich `if ... else ...` von der (neuen) `switch`-Anweisung?

b) Was sagt ein sogenanntes Hoare-Tripel $\langle\varphi\rangle\, \mathtt{P}\, \langle\psi\rangle$ aus?

c) Was ist der Unterschied zwischen partieller und totaler Korrektheit?

Tutoraufgabe 2 (Programmierung):

In dieser Aufgabe geht es um die Ein- und Ausgabe in Java. Dafür soll die bereitgestellte Klasse `SimpleIO` genutzt werden. Um einen String `str1` in einem Fenster mit dem Titel[1] `str2` auszugeben, nutzen Sie `SimpleIO.output(str1, str2)`. Um einen Wert vom Typ `type` mit der Klasse `SimpleIO` einzulesen, nutzen Sie `SimpleIO.getType(str)`, wobei `str` der Text ist, der der*dem Benutzer*in im Eingabefenster angezeigt wird. Um einen Wert vom Typ `int` einzulesen, benutzen Sie also z.B.
`SimpleIO.getInt("Bitte eine ganze Zahl eingeben")`.

Schreiben Sie ein einfaches Java-Programm, welches den*die Benutzer*in auffordert, eine positive ganze Zahl (d. h. größer als 0) einzugeben. Danach soll das Programm die Zahl einlesen. Diese Eingabeaufforderung mit anschließendem Einlesen soll solange wiederholt werden, bis der*die Benutzer*in wirklich eine positive Zahl eingibt. Wenn die Eingabe keine Zahl ist, darf sich das Programm beliebig verhalten. Anschließend soll der*die

[1] Sie dürfen in dieser Aufgabe immer z.B. "MultiEcho" als Titel verwenden.

Benutzer*in aufgefordert werden, ein Wort einzugeben. Das Wort soll eingelesen und schließlich so oft hintereinander geschrieben ausgeben werden, wie durch die eingegebene positive Zahl festgelegt wurde.
Ein Ablauf des Programms könnte z.B. so aussehen:

```
Bitte geben Sie eine positive Zahl ein
0
Bitte geben Sie eine positive Zahl ein
3
Bitte geben Sie ein Wort ein
Programmierung
ProgrammierungProgrammierungProgrammierung
```

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.

Tutoraufgabe 4 (Verifikation):

Gegeben sei folgendes Java-Programm über den Integer-Variablen `x`, `y`, `z` und `r`:

$\langle 0 \leq \texttt{x} \wedge 0 < \texttt{y} \rangle$ (Vorbedingung)

```
z = 0;
r = x;
while (r >= y) {
   r = r - y;
   z = z + 1;
}
```

$\langle \texttt{z} = \texttt{x} \text{ div } \texttt{y} \rangle$ (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 $\mathbb{Z}$.
- div steht für die Integer-Division, d.h. 5 div 3 ergibt z.B. 1.
- 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.

$\langle 0 \leq \texttt{x} \wedge 0 < \texttt{y} \rangle$

$\langle$ _____ $\rangle$

```
z = 0;
```

$\langle$ _____ $\rangle$

```
r = x;
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
while (r >= y) {
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
    r = r - y;
```

$\langle$ _____ $\rangle$

```
    z = z + 1;
```

$\langle$ _____ $\rangle$

```
}
```

$\langle$ _____ $\rangle$

$\langle \texttt{z} = \texttt{x} \text{ div } \texttt{y} \rangle$

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;
   }
```

$\langle$`res` $= \lceil\sqrt{\texttt{x}}\,\rceil\rangle$ (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.

$\langle \texttt{x} > 0 \rangle$

$\langle$ _____ $\rangle$

```
res = 1;
```

$\langle$ _____ $\rangle$

```
s = 1;
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
while (s < x) {
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
    res = res + 1;
```

$\langle$ _____ $\rangle$

```
    s = s + 2 * res - 1;
```

$\langle$ _____ $\rangle$

```
}
```

$\langle$ _____ $\rangle$

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

Tutoraufgabe 6 (Verifikation):

Gegeben sei folgendes Java-Programm P über den Integer-Variablen `n`, `i` und `res`:

$\langle 0 \leq \texttt{n} \rangle$ (Vorbedingung)

```
   i = 0;
   res = 0;
   while (i < n) {
      if (i % 2 == 0) {
         res = res + n;
      } else {
         res = res - 1;
      }
      i = i + 1;
   }
```

$\langle \texttt{res} = \lceil \frac{\texttt{n}}{2} \rceil \cdot \texttt{n} - \lfloor \frac{\texttt{n}}{2} \rfloor \rangle$ (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 $\mathbb{Z}$.
- 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.
- $\lceil x \rceil$ ist die kleinste Zahl $n \in \mathbb{Z}$, sodass $n \geq x$ gilt. $\lfloor x \rfloor$ ist die größte Zahl $n \in \mathbb{Z}$, sodass $n \leq x$ gilt.

$\langle 0 \leq \mathtt{n} \rangle$

$\langle$_____$\rangle$

```
i = 0;
```

$\langle$_____$\rangle$

```
res = 0;
```

$\langle$_____$\rangle$

```
while (i < n) {
```

$\langle$_____$\rangle$

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

$\langle$_____$\rangle$

$\langle$_____$\rangle$

```
        res = res + n;
```

$\langle$_____$\rangle$

```
    } else {
```

$\langle$_____$\rangle$

$\langle$_____$\rangle$

```
        res = res - 1;
```

$\langle$_____$\rangle$

```
    }
```

$\langle$_____$\rangle$

```
    i = i + 1;
```

$\langle$_____$\rangle$

```
}
```

$\langle$_____$\rangle$

$\langle \mathtt{res} = \lceil \frac{\mathtt{n}}{2} \rceil \cdot \mathtt{n} - \lfloor \frac{\mathtt{n}}{2} \rfloor \rangle$

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.

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

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

$\langle \texttt{b} \geq 0 \rangle$ (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;
      }
   }
```

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

$\langle b \geq 0 \rangle$

$\langle \_\_\_\_\_ \rangle$

```
x = b;
```

$\langle \_\_\_\_\_ \rangle$

```
res = a;
```

$\langle \_\_\_\_\_ \rangle$

```
y = 1;
```

$\langle \_\_\_\_\_ \rangle$

$\langle \_\_\_\_\_ \rangle$

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

$\langle \_\_\_\_\_ \rangle$

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

$\langle \_\_\_\_\_ \rangle$

$\langle \_\_\_\_\_ \rangle$

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

$\langle \_\_\_\_\_ \rangle$

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

$\langle \_\_\_\_\_ \rangle$

```
  } else {
```

$\langle \_\_\_\_\_ \rangle$

$\langle \_\_\_\_\_ \rangle$

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

$\langle \_\_\_\_\_ \rangle$

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

$\langle \_\_\_\_\_ \rangle$

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

$\langle \_\_\_\_\_ \rangle$

```
  }
```

$\langle \_\_\_\_\_ \rangle$

```
}
```

$\langle \_\_\_\_\_ \rangle$

$\langle res = a + b \rangle$

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.

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.