Informatik I **Wintersemester 2018/19**
Prof. Dr. Carsten Damm Georg-August-Universität Göttingen
Dr. Henrik Brosenne Institut für Informatik

Übungsblatt 06

E-Learning

Absolvieren Sie die Tests bis Di., 04.12., 23:55 Uhr.

Die Tests sind in der Stud.IP-Veranstaltung *Informatik I* unter *Lernmodule* hinterlegt.

Sie können einen Test **nur einmal durchlaufen**. Sobald Sie einen Test starten steht Ihnen nur eine **begrenzte Zeit** zu Verfügung, um den Test zu bearbeiten.

Alle Punkte, die Sie beim Test erreichen, werden ihnen angerechnet.

ILIAS – 5 Punkte

Felder

Absolvieren Sie den Test *Informatik I - ILIAS 06 Teil 1.*
(5 Punkte)

Hinweis

Wenn Sie den Test einmal vollständig durchlaufen haben kommen Sie auf die Seite *Testergebnisse*. Starten Sie den Test erneut aus Stud.IP, ist jetzt auch eine Schaltfläche *Testergebnisse anzeigen* vorhanden, die auf diese Seite führt.

Auf der Seite *Testergebnisse* können Sie sich unter *Übersicht der Testdurchläufe* zu jedem Testdurchlauf *Details anzeigen* lassen.

In der Auflistung der Aufgaben führt der Titel einer Aufgabe zu einer **Musterlösung** für die jeweilige Aufgabe.

ILIAS 4-Minuten-Aufgaben – 12 Punkte

Absolvieren sie die Tests *Informatik I - ILIAS 06 Teil 2, Teil 3* und *Teil 4.*
(12 Punkte)

LON-CAPA – 12 Punkte

Binärbruch, Schleifen

Absolvieren Sie im Test *Informatik I - LON-CAPA* die *Übung 06.*
(12 Punkte)

Übung

Abgabe bis Di., 04.12., 18 Uhr.

Werfen Sie Ihre Lösung in den Zettelkästen Ihrer Gruppenübung. Für die Übungen im Nordbereich stehen die Zettelkästen im Sockelgeschoß (Ebene -1) **oder** auf dem Flur vor dem Seminarraum auf Ebene 0 des Instituts für Informatik.

Wenn Ihre Übung im Südbereich stattfindet, klären Sie mit Ihrem Tutor wo die Lösungen abzugeben sind.

Achten Sie darauf, dass Ihr **Name**, Ihre **Gruppe** und Ihre **Matrikelnummer** auf **jedem** Blatt stehen!

Falls Ihre Lösung mehrere Blätter umfasst, heften Sie diese bitte zusammen.

Aufgabe 1 – 6 Punkte

Java. Type Casting

Betrachten Sie folgende ausführbare Klasse.

```
public class TypeCasting {
    public static void main(String[] args) {
        short  a=73;
        short  b=219;
        short  c=(short) (a+b);
        int    d=(byte) c;
        short  e=-3;
        int    f=(char) e;
        int    g=1;
        int    h=4;
        double i=g/h*4.0;

        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
        System.out.println(d);
        System.out.println(e);
        System.out.println(f);
        System.out.println(g);
        System.out.println(h);
        System.out.println(i);
    }
}
```

1. Welche Ausgabe produziert die `main`-Methode und warum.
   (4 Punkte)

2. Warum verlangt der Java-Compiler in Zeile 5 ein *type casting*.
   (2 Punkte)

Aufgabe 2 – 30 Punkte

Binärcode

Es sei $a = (a_0, a_1, \ldots, a_{n-1})$ eine n-stellige Binärziffernfolge mit $a_i \in \{0, 1\}$. Die Ableitung von a ist die n-stellige Binärziffernfolge $b = (b_0, b_1, \ldots, b_{n-1})$ mit $b_i \in \{0, 1\}$, deren Ziffern folgendermaßen entstehen.

$$\text{Für } i = 0, \ldots n-1 \text{ gilt} \qquad b_i = \begin{cases} 0 & \text{falls } i = 0 \text{ und } a_i = 0 \\ 0 & \text{falls } i > 0 \text{ und } a_i = a_{i-1} \\ 1 & \text{sonst} \end{cases}$$

1. Was stellt die Ableitung da? D.h. wenn die Ableitung gegeben ist, welche Aussagen kann man dann über die abgeleitete Binärziffernfolge treffen?
   (4 Punkte)

2. Gebenen Sie Java-Code einer rekursiven `void`-Methode an, die die Ableitung einer Binärziffernfolge bestimmt.

   Unter den Parametern der Methode befindet sich genau ein Feld mit Binärziffern. Dieses Feld wird mit jedem Methodenaufruf weitergereicht und während der Abarbeitung des Algorithmus mit der Ableitung überschrieben. Die Methode verwendet, neben dem übergebenen Feld, kein weiteres Feld.

   Wie sieht ein Aufruf der Methode aus?
   (9 Punkte)

   Hinweis. Durchlaufen Sie das Feld vom a_{n-1} nach a_0.

3. Formulieren Sie eine Vorschrift für die Umkehrung der Ableitung, d.h. eine Vorschrift, die angibt, wie man aus einer gegebenen Ableitung die abgeleitete Binärziffernfolge bestimmt.
   (8 Punkte)

4. Gebenen Sie Java-Code einer iterativen `void`-Methode an, die die Umkehrung einer Ableitung bestimmt, entsprechend der Vorschrift aus Aufgabenteil 3.

   Die Methode arbeitet auf der Eingabe, d.h. die übergebene Binärziffernfolge wird während der Abarbeitung des Algorithmus mit der Umkehrung überschrieben. Die Methode verwendet, neben der Eingabe, kein weiteres Feld.
   (9 Punkte)

   Hinweis. Durchlaufen Sie das Feld vom a_0 nach a_{n-1}.

Praktische Übung

Abgabe der Prüfsumme bis Di., 04.12., 23:55 Uhr.
Testat von Mi., 12.12., 8-10 Uhr bis Mo., 17.12., 18-20 Uhr.

Hilfe zum Bearbeiten der praktischen Übungen können Sie grundsätzlich jeden Tag in den Rechnerübungen bekommen, insbesondere in den Rechnerübungen am **Dienstag**, in denen **keine Testate** stattfinden.

Hinweise zu den praktischen Übungen, insbesondere zur **Abgabe der Prüfsumme** und zur **Durchführung der Testate**, sind in der Stud.IP-Veranstaltung *Informatik I* unter *Dateien→Übungsblätter* hinterlegt.

Aufgabe 1 – 35 Punkte

Collatz-Vermutung

Für eine positive ganze Zahl n kann mit nachfolgendem Bildungsgesetz eine Folge von positiven ganzen Zahlen erzeugte werden. Es gilt $a_0 = n$ und für $i > 0$ gilt

$$a_i = \begin{cases} \frac{1}{2}a_{i-1} & \text{wenn } a_{i-1} \text{ gerade ist,} \\ 3a_{i-1} + 1 & \text{wenn } a_{i-1} \text{ ungerade ist.} \end{cases}$$

Die Collatz-Vermutung besagt, dass es für jede positive ganze Zahl n ein $i > 0$ gibt mit $a_i = 1$.
Beispiele

- $n = 4$
  $a_0 = 4, a_1 = 2, a_2 = 1$
- $n = 13$
  $a_0 = 13, a_1 = 40, a_2 = 20, a_3 = 10, a_4 = 5, a_5 = 16, a_6 = 8, a_7 = 4, a_8 = 2, a_9 = 1.$

Schreiben Sie eine Applikation mit folgender Funktionsweise

- Zu einer beliebigen positiven ganzen Zahl n wird die oben definierte Zahlenfolge ausgegeben.
- Für zwei positive ganzen Zahlen n und m, für die gilt $n \leq m$, kann bewiesen werden, dass die Collatz-Vermutung für alle Zahlen im Intervall $[n, m]$ zutrifft.

Setzen Sie die Anforderungen wie folgt um.

1. Implementieren Sie eine Methode

```
static void collatzIterativOutput(int n)
```

die iterativ (nicht rekusiv) die Glieder der oben definierten Folge berechnet und ausgibt (jeweils eine Zeile mit Index und Wert des Folgenglieds getrennt durch ein Leerzeichen). Es werden solange neue Folgenglieder berechnet bis das erste Folgenglied mit Wert 1 auftritt oder `Integer.MAX_VALUE` viele Folgenglieder berechnet wurden oder bei der Berechnung eines Folgenglieds ein Wert größer `Integer.MAX_VALUE` auftreten würde.
(10 Punkte)

2. Implementieren Sie eine Methode

```
static int [] collatzIterativ(int n)
```

die iterativ die Glieder der oben definierten Folge berechnet und ein Feld zurückliefert, dass an Index i das Folgenglied a_i enthält. Tritt einer der in Aufgabenteil 1. beschriebenen Fehler auf wird `null` zurückgeliefert.

Implementieren Sie dazu ein Hilfsmethode

```
public static int[] appendInt(int[] a, int b)
```

die ein Feld zurückliefert, das mit den Elemente des Feldes `a` beginnt und zusätzlich noch das Element `b` enthält.

Verwenden Sie in `collatzIterativ` die Methode `appendInt`, um das Feld, das zurückgeliefert werden soll, schrittweise zu füllen.
(10 Punkte)

3. Implementieren Sie eine `main`-Methode, die entweder eine Zahl `n` oder zwei ganze Zahlen `n` und `m` von der Kommandozeile übernimmt.

   Werde auf der Kommandozeile weniger als eine oder mehr als zwei Zahlen angegeben, wird eine Fehlermeldung ausgegeben und die Applikation beendet. Sind `n` oder `m` keine positiven ganzen Zahlen oder gilt `n` größer `m` wird eine Fehlermeldung ausgegeben und die Applikation beendet.

   a) Wird eine positive ganze Zahl übergeben wird die Methoden `collatzIterativOutput`, sowie die Methode `collatzIterativ` aufgerufen und das von `collatzIterativ` zurückgelieferte Feld wird ausgegeben.

   b) Werden zwei positive ganze Zahlen `n` und `m` übergeben, dann wird mit der Methode `collatzIterativ` geprüft, ob die Collatz-Vermutung für die Zahlen `n` bis `m` zutrifft oder ob darüber keine Aussage gemacht werden kann.

   (10 Punkte)

4. Versehen Sie die Klasse mit ausführlichen Kommentaren, die den Programmablauf erläutern.
   (5 Punkte)