Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Peer Kröger
Michael Fromm, Florian Richter

München, 03.11.2016

Einführung in die Programmierung

WS 2018/19

Übungsblatt 3: Vollständige Induktion, p-adische Zahlendarstellung

Besprechung: 12.11 - 16.11.2018

Aufgabe 3-1 *Vollständige Induktion mit Ungleichungen*

(a) Zeigen Sie: $n^2 > 2n + 1$ für alle $3 \leq n \in \mathbb{N}$.

(b) Zeigen Sie: $2^n > n^2$ für alle $5 \leq n \in \mathbb{N}$ (Sie dürfen 3-1-a hierfür als bewiesen annehmen).

(c) Zeigen Sie, dass für alle $x \in \mathbb{R}, x \geq -1$ und für alle $n \in \mathbb{N}$ gilt: $(1 + x)^n \geq 1 + nx$

Hinweis: Da sie hier keine Gleichungen zeigen müssen, ist es manchmal notwendig, die Terme geschickt zu erweitern oder zu verringern.

Aufgabe 3-2 *Fehlerhafte Induktion*

Einer Ihrer Mitstudenten hat per Email Folgendes behauptet: *Wenn Einer die EIP-Klausur besteht, dann auch alle anderen!*
Auf Nachfragen folgte die Erklärung des Studenten als Induktionsbeweis:
(Induktionsanfang) *Angenommen, nur ein Student schreibt die Klausur mit und besteht. Dann haben offensichtlich alle Mitschreibenden bestanden.*
(Induktionsvoraussetzung) *Nun nehmen wir mal an, dass für $n \in \mathbb{N}$ Studenten S_n, die Behauptung gilt: Wenn ein Student $\tilde{s} \in S_n$ besteht, dann bestehen auch alle anderen $s \in S_n \setminus \{\tilde{s}\}$.*
(Induktionsschritt) *Schließlich meldet sich noch ein weiterer Student an. Die Menge der Studenten S_{n+1} hat nun eine Größe von $n + 1$. Wir nehmen an, dass ein Student $\tilde{s} \in S_{n+1}$ die Klausur besteht. Wir können S_{n+1} in zwei sich überschneidende Mengen von Studenten aufteilen:*

$$S_{n+1} = \{s_1, \ldots, s_{n+1}\} = \{s_1, s_2, \ldots, s_n\} \cup \{s_2, s_3, \ldots, s_{n+1}\} =: S' \cup S''$$

Der Student $\tilde{s}$, der nach Annahme besteht, ist in mindestens einer der Mengen S' oder S''. Die erste Menge $S' = s_1, \ldots, s_n$ sind die n zuerst angemeldeten Studenten. Vermutlich ist $\tilde{s} \in S'$, weil er immer brav in die Vorlesung ging und alle Informationen bezüglich Klausuranmeldungen mitbekommen hat. Nach Anwendung der Induktionsvoraussetzung bestehen dann alle $s \in S'$. Die zweite Menge sind die zuletzt angemeldeten Studenten $S'' = \{s_2, \ldots, s_{n+1}\}$. Das sind ebenfalls n viele, von denen die $n - 1$ Studenten $\{s_2, \ldots, s_n\} = S' \cap S''$ ja schon bestanden haben. Also kann auch hier die Induktionsvoraussetzung angewendet werden (falls $\tilde{s}$ sich zuletzt angemeldet hat, dann hat auf jeden Fall S'' bestanden. Dann aber auch S'). Folglich bestehen alle $n + 1$ Studenten.
Damit folgt, dass wir alle bestehen, weil einer schafft es doch immer.

Begründen Sie, warum dieser Student die vollständige Induktion noch einmal nachschlagen sollte.

Aufgabe 3-3 *p-adische Zahlendarstellungen*

Die Datei `p-adisch.txt` enthält folgende Tabelle:

$p = 2$	$p = 8$	$p = 10$	$p = 16$
1101	15	13	D
	76		
		26	
1111001			
			7C

Ergänzen Sie die Tabelle so, dass in jeder Zeile die verschiedenen p-adischen Zahlendarstellungen für die selbe Zahl stehen.

Aufgabe 3-4 *Karten umdrehen*

Gegeben ist das folgende Ein-Spieler-Spiel. Vor Ihnen liegen $n \in \mathbb{N}$ Spielkarten verdeckt in einer Reihe nebeneinander auf dem Tisch. Ein Zug besteht immer aus zwei Schritten:

1. Eine beliebige verdeckte Karte wird ausgewählt und herumgedreht, sodass ihre Bildseite zu sehen ist.

2. Die Karte, die direkt rechts neben der gewählten Karte liegt, wird ebenfalls herumgedreht - ungeachtet dessen, ob sie noch verdeckt ist oder nicht. Gibt es dort keine Karte, dann überspringe diesen Schritt.

(a) Terminiert das Spiel nach endlich vielen Zügen? Begründen Sie schlüssig Ihre Antwort.

(b) Angenommen, in jedem Zug muss die Karte links von der gewählten Karte statt der rechten Nachbarkarte umgedreht werden. Ändert dies etwas? Begründen Sie.

(c) Nun erlauben wir, dass sich der Spieler in jedem Zug aussuchen darf, ob er im 2. Schritt die linke oder die rechte Karte umdreht. Was können Sie nun über das Terminierungsverhalten sagen?

Hinweis: Sie können die binäre Zahlendarstellung als Repräsentation der Kartenreihe nutzen .

Aufgabe 3-5 *Java/Benutzerinteraktion*

In Aufgabe 0-3 wurde ein Programm kompiliert, das eine feste Ausgabe besaß. Die meisten Programme nutzen aber unterschiedliche Eingaben, um darauf zu reagieren und spezifische Ausgaben zu erzeugen. Dazu benötigen wir eine Benutzereingabe. Es gibt einige Möglichkeiten, dies in Java zu realisieren. Vorerst nutzen wir die Scannerklasse, die einige Eingabemethoden zur Verfügung stellt.

(a) Erstellen Sie wie in Aufgabe 0-3 eine .java-Datei mit dem Namen `Eingabe.java`. Geben Sie dieser Datei vorerst das Hello-World-Programm als Inhalt. Kompilieren Sie das Programm. Folgende Fehlermeldung wird erscheinen:

```
Eingabe.java:3: error: class HelloWorld is public,
should be declared in a file named HelloWorld.java
public class HelloWorld {
       ^
1 error
```

Java-Dateien enthalten immer genau eine öffentliche Klasse (public class). Ändern Sie daher den Namen der Klasse von `HelloWorld` zu `Eingabe`. Ändern Sie außerdem die Ausgabe von `"Hello World"` zu `"Wie alt bist du: "`.

(b) Nach der Ausgabe soll eine ganze Zahl eingelesen werden. Dazu importieren Sie mittels `import java.util.Scanner;` am Dateianfang eine Klasse, die Ihnen dabei hilft, Texteingaben zu verarbeiten. Danach nutzen Sie `Scanner sc = new Scanner(System.in);`, um dem Programm zu erklären, dass es einen Scanner vorbereiten soll, der Eingaben von der Eingabekonsole (`System.in`) erwartet. Dieser heißt `sc`. Anschließend soll der Scanner eine ganze Zahl des Benutzers einlesen: `int alter = sc.nextInt();`. Geben Sie auf geeignete Weise die Eingabe in der Konsole aus, z.B.: `"Dein Alter ist 13."`. Das Grundgerüst für diese Aufgabe liegt als `Eingabe.java` zum Download bereit.

(c) Nun reagieren Sie altersgerecht auf die Eingabe. Personen mit einem Alter über oder gleich 16 haben Zugriff auf Bier. Allen anderen wird dies verweigert. Dazu brauchen wir ein Konstrukt zur Abfrage von Bedingungen. Dabei gibt es im Programm eine Verzweigung, deren erster Block nur durchlaufen wird, wenn die Bedingung erfüllt ist. Ansonsten wird entweder das Konstrukt ganz übersprungen, oder der zweite Block durchgearbeitet. Im folgenden sehen Sie die Umsetzung in Java:

```
if(Bedingung) {
        //Wenn Bedingung erfuellt ist, dann tue alles in diesem Block.
}
else {
        //Wenn Bedingung nicht erfuellt ist, dann tue dies hier.
}
```

Formulieren Sie eine geeignete Bedingung zur Altersabfrage und geben Sie in beiden Anweisungsblöcken eine entsprechende Ausgabe. Beispielausgaben können wie folgt aussehen:

```
Wie alt bist du: 13
Kein Bier fuer dich.

Wie alt bist du: 28
Hier ist dein Bier.
```

(d) Für die Motivierten: Fragen Sie bei positiver Altersabfrage nach, wieviel Biere n gewünscht sind, und geben sie dann n-mal das Wort Bier aus. Falls n negativ ist, verweigern Sie die Bierausgabe. Dazu benötigen Sie zusätzlich zum `if`-Konstrukt noch eine `while`-Schleife.