

Informatik I: Einführung in die Programmierung

Prof. Dr. Bernhard Nebel
Dr. Stefan Wöfl, Thorsten Engesser, Tim Schulte
Wintersemester 2016/2017

Universität Freiburg
Institut für Informatik

Übungsblatt 3

Abgabe: Freitag, 11. November 2016, 20:00 Uhr

WICHTIGE HINWEISE: Zur Bearbeitung der Übungsaufgaben legen Sie bitte ein neues Unterverzeichnis `sheet03` im Wurzelverzeichnis Ihrer Arbeitskopie des SVN-Repositories an. Ihre Lösungen werden dann in Dateien in diesem Unterverzeichnis erwartet.

Beachten Sie bitte bei allen Aufgaben die *Hinweise zur Bearbeitung der Übungsaufgaben* unter der folgenden URL:

<http://gki.informatik.uni-freiburg.de/teaching/ws1617/info1/guide/hinweise.html>

Bewertet wird bei allen Aufgaben die letzte Version, die zur Deadline des Übungsblattes auf dem SVN-Server eingereicht ist.

Aufgabe 3.1 (Größter gemeinsamer Teiler; Datei: `gcd.py`, Punkte: 4+4)

In dieser Aufgabe soll ein Programm zur Berechnung des größten gemeinsamen Teilers zweier oder mehrerer Zahlen implementiert werden. Benutzen Sie hierzu das auf der Übungsseite bereitgestellte Template.

- Implementieren Sie die Funktion `gcd(m, n)`, die den größten gemeinsamen Teiler zweier positiver, natürlicher Zahlen `m` und `n` (jeweils vom Typ `int`) berechnet. Verwenden Sie dazu die iterative Version des euklidischen Algorithmus (https://de.wikipedia.org/wiki/Euklidischer_Algorithmus#Iterative_Variante). Es dürfen keine zusätzlichen Python-Module importiert werden.
- Implementieren Sie die Funktion `cli_gcd()`, die beliebig viele positive, natürliche Zahlen mittels `input` von der Standardeingabe einliest und den größten gemeinsamen Teiler dieser Zahlen mittels `print` auf der Standardausgabe ausgibt. Ihr Programm soll so lange neue Zahleneingaben akzeptieren, bis eine 0 eingegeben wurde, die dann für die Berechnung des größten gemeinsamen Teilers nicht mehr berücksichtigt wird.

Wenn Sie diese Funktionen in IDLE implementieren, könnte ein Aufruf der Funktion in der IDLE-Shell (bei entsprechenden Nutzereingaben) wie folgt aussehen:

```
>>> cli_gcd()
Calculate GCD from the input natural numbers.
Input nat. number (or 0 to quit): 294
Input nat. number (or 0 to quit): 84
Input nat. number (or 0 to quit): 126
Input nat. number (or 0 to quit): 0
GCD: 42
>>>
```

Hinweis: Sie dürfen bei der Bearbeitung dieser Aufgabe davon ausgehen, dass die Nutzereingaben sich stets mittels der `int`-Funktion in einen `int`-Wert konvertieren lassen.

Aufgabe 3.2 (Automaten; Dateien: (a) `automaton-diagram.jpg` oder `.pdf`, (b) `automaton.py`, (c) `automaton-test.txt`, (d) `automaton2.txt`; Punkte: 2+5+2+1)

Wir wollen einen Getränkeautomaten als Moore-Automaten implementieren. Wir nehmen an, dass der Getränkeautomat lediglich 50-Cent- und 1-Euro-Münzen annimmt. Ab einem Guthaben von 2,00 € werden keine weiteren Münzen mehr angenommen. Wird ein Getränk gewählt, das weniger kostet als das momentan vorhandene Guthaben, bleibt der Restbetrag erhalten (es gibt keine Geldrückgabe). Die folgenden Getränke stehen zur Verfügung: Cola für 1,50 € und Energy Drink für 2,00 €. Die Getränke können erst ausgewählt werden, wenn bereits entsprechendes Guthaben vorhanden ist. Wir nehmen an, dass Getränke unbeschränkt zur Verfügung stehen.

Für die Modellierung des Getränkeautomaten als Moore-Automaten verwenden wir insgesamt vier Eingabesymbole: `inp50` und `inp100` für den Einwurf von 50ct bzw. 1 Euro, sowie `reqCoke` und `reqEnergy` für die Getränkeauswahl. Als Ausgabealphabet verwenden wir die Symbole `0ct`, `50ct`, `100ct`, `150ct`, `...`, die beim Einwurf eines Geldstücks ausgegeben werden um das aktuelle Guthaben anzuzeigen, sowie die Symbole `COKE` und `ENERGY` zur Rückgabe des Getränks. Beachten Sie dabei Groß- und Kleinschreibung.

- (a) Formalisieren Sie den oben beschriebenen Automaten. Überlegen Sie dazu zunächst, in welchen Zuständen der Automat sein kann und welche Nachfolgezustände jeder Zustand hat. Zeichnen Sie dann das Übergangsdiagramm für den Moore-Automaten.

Sie dürfen das Diagramm von Hand anfertigen und abfotografieren/einscannen oder auch gerne ein dafür geeignetes Zeichenprogramm verwenden. Erlaubte Dateiformate für die Abgabe sind ausschließlich JPEG- und PDF-Dateien.

- (b) Implementieren Sie den Automaten in Python analog zu der Implementierung des Würfelautomaten aus der Vorlesung. Benutzen Sie dazu das auf der Übungsseite bereitgestellte Template. Eingaben und Ausgaben werden dabei direkt auf der Konsole getätigt. Ein möglicher Testlauf könnte wie folgt aussehen:

```
>>> import automaton
>>> automaton()
> inp50
50ct
> inp100
150ct
> inp100
250ct
> inp50
250ct
> reqCoke
COKE
> inp50
150ct
>
```

- (c) Führen Sie einen ausführlichen Testlauf Ihres Automaten durch, der mittels drei aufeinanderfolgender Getränkeausgaben die Funktionsweise des Automaten aufzeigt. Speichern Sie den Testlauf in der Datei `automaton-test.txt`.
- (d) Überlegen Sie sich, wie Sie den Automaten anpassen müssten, um die zusätzliche Funktionalität einer Geldrückgabetaaste zu implementieren. Wie viele neue Zustände müssten Sie einführen und wie würde die Übergangsfunktion dann aussehen? Abgabe in `automaton2.txt`.

Aufgabe 3.3 (Erfahrungen; Datei: `erfahrungen.txt`; Punkte: 2)

Legen Sie im Unterverzeichnis `sheet03` eine Textdatei `erfahrungen.txt` an. Notieren Sie in dieser Datei kurz Ihre Erfahrungen beim Bearbeiten der Übungsaufgaben (Probleme, benötigter Zeitaufwand nach Teilaufgabe, Interessantes, etc.).