Dr. Steffen Jost
Christian Neukirchen

Ludwig-Maximilians-Universität München
Institut für Informatik
12. April 2018

01. Übung zur Vorlesung

Programmierung und Modellierung

Das Übungsblatt momentan immer ab Donnerstag per UniWorX (`uniworx.ifi.lmu.de`) erhältlich. Wir gehen davon aus, dass während der Übung je 2–3 Teilnehmer gemeinsam einen Laptop mit installiertem GHC zur Verfügung haben. Hinweise zur Installation von GHC finden Sie auf der Vorlesungshomepage (`www.tcs.ifi.lmu.de/lehre/ss-2018/promo/`). Für Teilnehmer ohne Laptops gibt es eine Übung im CIP-Pool der Informatik.

Die mit A gekennzeichneten Aufgaben sind explizit zur Diskussion innerhalb von kleinen Gruppen gedacht. Rufen Sie den Tutor erst, wenn Ihre Gruppe wirklich stecken bleibt! Die mit H gekennzeichneten Aufgaben sind Hausaufgaben. Die meisten Hausaufgaben sind mit Punkten gekennzeichnet. Mit diesen Punkten kann ein Bonus auf die Klausurnote erzielt werden. Der Bonus beeinflusst aber nicht, ob man bestanden hat oder durchgefallen ist.

Falls Sie keinen eigenen Rechner haben, können Sie den CIP-Pool der Informatik nutzen. Die Frist zur Abgabe Ihrer Lösungen per UniWorX finden Sie immer am Ende des Übungsblattes. Bitte beachten Sie jeweils den in der Titelzeile der Aufgabe spezifizierten **Dateinamen und das Dateiformat**! Pro Übungsblatt kann nur ein ZIP-Archiv hochgeladen werden. Ein erneutes Hochladen ersetzt Ihre vorherige Abgabe komplett. Es werden ausschließlich `.zip`-Archive korrigiert, andere Archiv-Formate wie etwa `.rar`, `.tar`, `.gz`, etc. werden ignoriert! Es gibt immer Punktabzug, wenn eine Abgabe zu einer Programmieraufgabe nicht kompiliert; wenn der Dateiname nicht den Anforderungen entspricht; wenn das Dateiformat nicht stimmt. Diese Abzüge sollten leicht zu vermeiden sein!

Die Korrektur Ihrer Hausübungen erhalten Sie ungefähr eine Woche nach Ablauf der Abgabfirst ebenfalls über UniWorX. Bitte haben Sie dabei etwas Geduld! Da die Musterlösung automatisch nach Ablauf der Abgabefrist freigeschaltet wird, können wir leider keine nachträglichen Abgaben akzeptieren. Falls trotz Musterlösung und Korrektur immer noch Fragen offen bleiben, so wenden Sie sich bitte an Ihren Tutor oder nutzen Sie die Feedback Funktion zu Ihrer Korrektur in UniWorX. Bitte in jedem Fall die von UniWorX zugewiesene Nummer der Abgabe angeben!

A1-1 ***Kartesisches Produkt*** Berechnen Sie mit Papier und Bleistift die vollständige Menge des jeweiligen kartesischen Produktes. Wenn Sie möchten, können Sie ihre Antworten mit GHCi überprüfen.

Hinweis: im Gegensatz zu Listen ist die Reihenfolge der Elemente einer Menge unerheblich (und Mengen enthalten auch keine "doppelten" Elemente).

a) $\{11, 7, 3\} \times \{\spadesuit, \heartsuit\}$

b) $(\{1, 2\} \times \{3, 4\}) \times \{5, 6\}$

c) $\{1, 2\} \times (\{3, 4\} \times \{5, 6\})$

d) $\{1, 2\} \times \{\}$

e) Sei A eine Menge mit n Elementen. Wie viele Elemente gibt es in $A \times \{27, 69\}$?

A1-2 ***Auswertung*** Berechnen Sie möglichst mit Papier und Bleistift den Wert, zu dem der gegebene Ausdruck jeweils auswertet. Überprüfen Sie Ihr Ergebnis mit GHCi.

a) `'c':'o':'o':'l':"!"`

b) `(4 == 5.0):[]`

c) `let mond="käse" in if mond=="käse" && 1==2 then False else 1+1==2`

d)
```
[ z | c <- "grotesk", c/='k', c/='r', c/='e' && c/='s'
      , let z = if c=='o' then 'u' else c ]
```

e) `[(a,b) | a<-[1..5], b<-[5..1], a/=b]`

f) `(\x->"nope!") [c | c <- "yes!"]`

A1-3 ***GHCI Grundlagen*** Speichern Sie folgenden Code in eine Datei `mystery.hs`:

```
foo :: Bool -> (Bool -> Bool)
foo  True  False  =  False
foo  _     _      =  True
```

Starten Sie nun GHCi und laden Sie dort die Datei `mystery.hs`. Werten Sie nun die Funktion `foo` mehrfach mit verschiedenen Argumenten aus. Erstellen Sie daraus eine Wahrheitstafel mit allen möglichen Eingaben. *Bonusfrage:* Welchen logischen Operator implementiert `foo`?

A1-4 ***Pattern-Matching*** Schreiben Sie eine Funktion `myAnd`,[1] welche den Typ `Bool -> (Bool -> Bool)` hat und die logische Operation "und" implementiert. Zur Übung der Vorlesungsinhalte sollen Sie diese Aufgabe gleich drei Mal lösen, jeweils mit einer anderen Einschränkung:

a) Verwenden Sie bei der Definition ausschließlich Pattern-Matching.

b) Verwenden Sie bei der Definition ausschließlich Wächter (keine konstanten Patterns).

c) Verwenden Sie weder Pattern-Matching noch Wächter.

Natürlich dürfen Sie in keinem Fall die Operatoren aus der Standardbibliothek verwenden!

[1] Zur Vermeindung von Namenkonflikten mit der Standardbibliothek nicht "and" verwenden

H1-1 ***Typen*** (0 Punkte; Keine Abgabe)
Erst im Kopf, dann mit GHCi überprüfen: Welchen Typ haben folgenden Ausdrücke?

a) `['a','b','c']`

b) `('a','b','c')`

c) `[(False,[1]),(True,[(2.0)])]`

d) `([True,True],('z','o','o'))`

e) `(\x -> ('a':x,False,([x])))`

f) `[(\x y-> (x*2,y-1)) m n | m <- [1..5], even m, n <- [6..10]]`

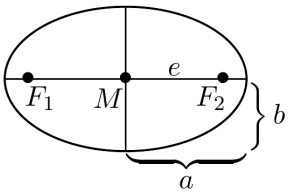
Hinweis: Kontrollieren Sie Ihre Antworten anschließend selbst mit GHCi! Auch wenn wir Typinferenz in der Vorlesung noch nicht behandelt haben, sollten Sie in der Lage sein, die meisten dieser einfachen Aufgaben bereits alleine mit Papier und Bleistift lösen zu können. Dies Aufgabe ist eigentlich auch einfacher als Aufgabe A1-2!

H1-2 ***Wahrheitstafeln*** (2 Punkte; Abgabe: `H1-2.txt` oder `H1-2.pdf`)

Was sind in Haskell die Symbole für logisches UND und logisches ODER?
Ermitteln Sie mit Hilfe von GHCi jeweils die Wahrheitstafel, ähnlich wie in Aufgabe A1-3!

H1-3 ***Modellieren mathematischer Funktion*** (2 Punkte; `.hs`-Datei abgeben)
Eine Ellipse um einen Mittelpunkt M ist gegeben durch ihre große Halbachse a und ihre kleine Halbachse b. Auf der großen Achse liegen die beiden Brennpunkte F_1, F_2 der Ellipse. Den Abstand e dieser Brennpunkte vom Mittelpunkt nennt man die Exzentrizität der Ellipse.

Die Exzentrizität hat einen Wert zwischen 0 und a und ist ein Maß dafür, wie „länglich“ die Ellipse ist. Die Exzentrizität hat den Wert 0 im Fall $a = b$, wenn also die Ellipse zu einem Kreis mit Radius a entartet. Die Exzentrizität hat den Wert a im Fall $b = 0$, wenn also die Ellipse zu einer Strecke der Länge $2a$ entartet.



Ellipse mit Halbachsen a und b mit $a \geq b \geq 0$	
Flächeninhalt:	πab
Exzentrizität:	$\sqrt{a^2 - b^2}$
Umfang:	$\pi\left(\frac{3}{2}(a+b) - \sqrt{ab}\right)$

In der obigen Tabelle sind die Formeln zur Berechnung von Flächeninhalt, Exzentrizität und Umfang angegeben.

Schreiben Sie eine Datei `H1-3.hs`, welche diese Funktionen implementiert:

```
flaecheninhalt :: Double -> Double -> Double
exzentrizitaet :: Double -> Double -> Double
umfang         :: Double -> Double -> Double
```

Hinweise

- Zur Vereinfachung müssen Sie nicht überprüfen, dass die Bedingung $a \geq b \geq 0$ tatsächlich eingehalten wird.
- Folgende Funktionen aus der Standardbibliothek dürfen Sie zusätzlich zu den üblichen mathematischen Operationen `+`,`-`,`*`,`/`,`^` verwenden: `pi`, `sqrt`. Schlagen Sie diese Funktionen ggf. in der Dokumentation der Standardbibliothek nach.
- *Beispiel* Eine Funktion zur Berechnung des Flächeninhalts eines Kreises, könnte man in Haskell wie folgt definieren:

  ```
  kreisfläche :: Double -> Double
  kreisfläche r = pi * r * r
  ```

- Punktabzug, falls Funktionsnamen, Signatur oder Dateiname nicht wie angegeben sind.

Abgabe: Lösungen zu den Hausaufgaben können bis Samstag, den 21.4.18, mit UniWorX nur als `.zip` abgegeben werden. Abschreiben bei den Hausaufgaben gilt als Betrug und kann zum Ausschluss von der Klausur zur Vorlesung führen. Bis zu 3 Studierende können gemeinsam als Gruppe abgeben. Bitte beachten Sie auch die Hinweise zum Übungsbetrieb auf der Vorlesungshomepage (`www.tcs.ifi.lmu.de/lehre/ss-2018/promo/`).