STARKER START ins Studium

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Fachbereich Informatik und Mathematik
Institut für Informatik

Vorsemesterkurs Informatik

Sommersemester 2018

Aufgabenblatt Nr. 4B

Aufgabe 1 (Haskell Interpreter: GHCi)

Starten Sie den Haskell Interpreter GHCi aus Ihrem Homeverzeichnis.

a) Verschaffen Sie sich einen Überblick über die Bedienung des GHCi, indem Sie sich die Hilfe im GHCi anzeigen lassen.

Lösung

`ghci` (oder falls nicht im `PATH` auf den RBI-Rechnern: `/opt/rbi/bin/ghci`) im Homeverzeichnis aufrufen und `:?` für die Hilfe im Interpreter.

b) Lassen Sie sich im GHCi das Verzeichnis anzeigen, in dem Sie sich befinden. Das Kommando `:!` ist hierbei hilfreich. Wechseln Sie anschließend in das Verzeichnis `~/vorkurs` *ohne* den GHCi zu verlassen.

Lösung

`:!pwd` und `:cd vorkurs`.

c) Geben Sie zu jedem der folgenden arithmetischen Ausdrücke den entsprechenden Haskell-Ausdruck an und lassen Sie den GHCi jeweils dessen Wert berechnen. Füllen Sie dabei die folgende Tabelle aus.

Arithmetischer Ausdruck	Ausdruck in Haskell	Ergebnis im GHCi
$1+3+5+7+9$		
$(15-6)\cdot 3+12\cdot 2$		
$\frac{1}{2}+\frac{1}{4}$		
$\left(\frac{1}{3}+\frac{1}{2}\right)+\frac{1}{6}$		
$\frac{1}{3}+\left(\frac{1}{2}+\frac{1}{6}\right)$		
$\frac{1}{0}$		
$-1\cdot 2$		
$2\cdot -1$		
$2^{2^{2^2}}$		

Hinweis: Verwenden Sie die Operatoren `*`, `/` und `^` zur Multiplikation, Division und Potenzierung.

Lösung

$1+3+5+7+9$	25
$(15-6)*3+12*2$	51
$1/2+1/4$	0.75
$(1/3+1/2)+1/6$	0.9999999999999999
$1/3+(1/2+1/6)$	1
$1/0$	$Infinity$
$-1*2$	-2
$2*(-1)$	-2
$2\hat{}2\hat{}2\hat{}2$	65536

Aufgabe 2 (Funktionalität testen)

Auf der Webseite zum Vorkurs (`http://vorkurs.informatik.uni-frankfurt.de/`) finden Sie eine Datei `magic.hs`. Laden Sie diese herunter und laden Sie sie anschließend in den GHCi. Die Datei stellt die Funktionen `fun1`, `fun2`, `fun3`, `fun4` und `fun5` bereit. Diese erwarten eine Zeichenkette als Eingabe und liefern eine veränderte Zeichenkette. Ein Test ist z.B. `fun1 "Hallo"`.

Finden Sie durch *Testen* der Funktionen `fun1` bis `fun5` heraus, welche der folgenden Änderungen diese Funktionen auf Zeichenketten durchführen (Mehrfachantworten pro Funktion sind möglich) und kreuzen Sie die entsprechenden Antworten an. Beachten Sie, dass der Quellcode der Datei `magic.hs` mit Absicht nahezu unverständlich ist.

Wirkung	`fun1`	`fun2`	`fun3`	`fun4`	`fun5`
1) macht aus Kleinbuchstaben Grossbuchstaben	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein
2) macht aus Grossbuchstaben Kleinbuchstaben	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein
3) macht aus dem i-ten Buchstaben des Alphabets die Zahl i	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein
4) erniedrigt alle Ziffern (ausser der 0) um 1	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein
5) macht aus jedem Fragezeichen ein Ausrufezeichen	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein
6) ersetzt alle Ziffern (außer 0), durch ihre Darstellung als römische Zahl	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein
7) entfernt alle runden Klammern	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein
8) verdoppelt alle Vorkommen der Buchstaben x,y,z,X,Y,Z	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein
9) vertauscht in Sätzen manche Worte	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein
10) entfernt Worte, die mehrfach im Text auftauchen	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein
11) löscht Leerzeichen, falls Worte mit mehr als einem Leerzeichen getrennt sind	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein	☐ Ja ☐ Nein

Lösung

fun1 4) 7)
fun2 1) 7) 10) 11)
fun3 3) 6)
fun4 8) 9) 11)
fun5 2) 4) 5)

Aufgabe 3 (Boolesche Ausdrücke und Rätsel lösen)

a) Überlegen Sie sich für die folgenden Booleschen Ausdrücke, *welchen* Wahrheitswert diese jeweils darstellen. Überprüfen Sie *anschließend* Ihr Ergebnis, indem Sie die Ausdrücke im GHCi auswerten lassen. Hierfür müssen Sie die Ausdrücke in Haskell-Notation überführen (d.h. `True` und `False` für die Wahrheitswerte und `not`, `&&`, `||` für $\neg$, $\wedge$, $\vee$ verwenden).

- ¬wahr
- wahr ∨ falsch
- ¬((falsch ∨ wahr) ∧ (¬falsch))
- ¬((falsch ∨ (¬falsch)) ∧ (¬wahr))
- ¬(¬((¬wahr) ∨ falsch))

Lösung

Im Skript, Abschnitt 3.2.1 finden sich z.B. Wahrheitstabellen und Erläuterungen.

- ¬wahr
  ergibt falsch
  Haskell:

```
Prelude> not True
False
```

- wahr ∨ falsch
  ergibt wahr
  Haskell:

```
Prelude> True || False
True
```

- ¬((falsch ∨ wahr) ∧ (¬falsch))
  ergibt ¬(wahr ∧ wahr)
  ergibt ¬wahr
  ergibt falsch
  Haskell:

```
Prelude> not((False || True) && (not False))
False
```

- ¬((falsch ∨ (¬falsch)) ∧ (¬wahr))
  ergibt ¬((falsch ∨ wahr) ∧ falsch)
  ergibt ¬(wahr ∧ falsch)
  ergibt ¬falsch
  ergibt wahr
  Haskell:

```
Prelude> not((False || (not False)) && (not True))
True
```

- ¬(¬((¬wahr) ∨ falsch)) *ergibt* ¬(¬(falsch ∨ falsch))
  ergibt ¬(¬falsch)
  ergibt ¬wahr
  ergibt falsch
  Haskell:

```
Prelude> not (not ((not True) || False))
False
```

b) Lügenbolde lügen immer, während Wahrsager immer die Wahrheit sagen. Jeder der drei Brüder Knasi, Knesi und Knosi ist entweder ein Lügenbold oder ein Wahrsager. Knasi sagt: “Wir Brüder sind alle Lügenbolde”. Knesi sagt: “Genauer einer von uns sagt die Wahrheit”.

Formalisieren Sie das Rätsel mit Aussagenlogik in Haskell, indem Sie zunächst definieren

```
knasi_ist_wahrsager = undefined
knesi_ist_wahrsager = undefined
knosi_ist_wahrsager = undefined
```

und anschließend die beiden Aussagen in Haskell formulieren und verknüpfen:

```
knasis_aussage = ...
knesis_aussage = ...

beide_aussagen = ...
```

Probieren Sie anschließend durch alle Belegungen (`True` oder `False`) für die Variablen `knasi_ist_wahrsager`, `knesi_ist_wahrsager`, `knosi_ist_wahrsager` aus (indem Sie `undefined` durch die Wahrheitswerte ersetzen) und lassen Sie den Wert von `beide_aussagen` berechnen.

Für welche Belegung(en) sind beide Aussagen wahr, d.h. wer ist Lügenbold und wer ist Wahrsager?

Lösung

```
    --  Knasi sagt: "Wir Brüder sind alle Lügenbolde".
knasis_aussage =
    -- 1. Möglichkeit: Knasis sagt die Wahrheit und alle drei sind L"ugenbolde
      (knasi_ist_wahrsager && not knasi_ist_wahrsager && not knesi_ist_wahrsager && not kno
    -- oder 2. Möglichkeit: Knasi l"ugt, und nicht alle sind L"ugenbolde
     || (not knasi_ist_wahrsager && not (not knasi_ist_wahrsager && not knesi_ist_wahrsager
    -- Knesi sagt: "Genauer einer von uns sagt die Wahrheit".
knesis_aussage =
    -- 1.Möglichkeit: Knesi sagt die Wahrheit und es gibt genau einen
      (knesi_ist_wahrsager && (knasi_ist_wahrsager || knesi_ist_wahrsager || knosi_ist_wahr
                           && not (knasi_ist_wahrsager && knesi_ist_wahrsager)
                           && not (knasi_ist_wahrsager && knosi_ist_wahrsager)
                           && not (knesi_ist_wahrsager && knosi_ist_wahrsager))
     ||
    -- 2.Möglichkeit: Knesi lügt und es gibt nicht genau einen
      (not knesi_ist_wahrsager && not ((knasi_ist_wahrsager || knesi_ist_wahrsager || knosi
                           && not (knasi_ist_wahrsager && knesi_ist_wahrsager)
                           && not (knasi_ist_wahrsager && knosi_ist_wahrsager)
                           && not (knesi_ist_wahrsager && knosi_ist_wahrsager)))

beide_aussagen = knasis_aussage && knesis_aussage


-- nur diese Belegung liefert wahr f"ur beide_aussagen
knasi_ist_wahrsager = False
knesi_ist_wahrsager = True
knosi_ist_wahrsager = False


-- eher lange L"osung, man kann optimieren, z.B. sieht man
-- bei knesis_aussage, dass wenn er die Wahrheit sagt, nur er der wahrsager ist usw.
```

Aufgabe 4 (Verzeichnisse)

a) Finden Sie heraus, wie Ihr Homeverzeichnis auf Ihrem Rechner, Ihr Username und der Name des Rechners, auf dem Sie arbeiten, heißen.

Lösung

```
echo ~
whoami
hostname
```

b) Legen Sie in Ihrem Homeverzeichnis die im folgenden Baum dargestellten Unterverzeichnisse ausschließlich unter Verwendung der Shell-Befehle `cd` und `mkdir` an, ohne dabei ein `/` zu benutzen.

```
vorkurs
 |-- verzeichnis-1
 |   |-- unterverzeichnis-1-1
 |   |-- unterverzeichnis-1-2
 |   `-- unterverzeichnis-1-3
 |-- verzeichnis-2
 |   `-- .verstecktes_unterverzeichnis-2-1
 `-- verzeichnis-3
     |-- unterverzeichnis-3-1
     `-- unterverzeichnis-3-2
```

Welche Kommandos haben Sie eingegeben?

c) Wechseln Sie in Ihr Homeverzeichnis und führen Sie den Befehl `ls` mit den untengenannten Optionen (auch mit allen Kombinationen) und dem Parameter `vorkurs` aus. Erläutern Sie die Effekte der unterschiedlichen Optionen (schauen Sie dazu in die Man Page von `ls` (mittels `man ls`)).

- `-a`
- `-l`
- `-R`

Lösung

- `-R`: listet rekursiv auch die Unterverzeichnisse auf
- `-a`: zeigt auch versteckte Dateien
- `-l`: Listenformat für die Ausgabe

d) Informieren Sie sich über die Verwendung des Befehls `rm`, indem Sie die Man Page dazu durchlesen (mittels `man rm`) und/oder im Internet danach suchen. Löschen Sie die angelegten Unterverzeichnisse von `vorkurs` mithilfe des Befehls `rm`.

Lösung

Im Verzeichnis `vorkurs`
```
rm -r verzeichnis-1
rm -r verzeichnis-2
rm -r verzeichnis-3
```
Oder kürzer (aber mit `*` sollte man vorsichtig umgehen):
```
rm -r verzeichnis-*
```

e) Erstellen Sie Unterverzeichnisse von `vorkurs` erneut jedoch unter Verwendung des Kommandos `mkdirhier` (studieren Sie vorher die dazugehörige Man Page). Löschen Sie anschließend die Unterverzeichnisse von `vorkurs` erneut.

Lösung

```
mkdirhier verzeichnis-1/unterverzeichnis-1-1
mkdir verzeichnis-1/unterverzeichnis-1-2
mkdir verzeichnis-1/unterverzeichnis-1-3
mkdirhier verzeichnis-2/.verstecktes-unterverzeichnis-2-1
mkdirhier verzeichnis-3/unterverzeichnis-3-1
mkdir verzeichnis-3/unterverzeichnis-3-2
```

oder für Spezialisten:

```
for x in 1 2 3; do mkdirhier verzeichnis-1/unterverzeichnis-1-$x; done
mkdirhier verzeichnis-2/.verstecktes-unterverzeichnis-2-1
for x in 1 2; do mkdirhier verzeichnis-3/unterverzeichnis-2-$x; done
```

Aufgabe 5 (Textdateien)

a) Legen Sie mithilfe eines Editors eine Datei namens `IrgendeinText.txt` im Verzeichnis `vorkurs` an, und schreiben Sie irgendeinen Text der mindestens 100 Zeilen besitzt und das Wort `"Informatik"` enthält in die Datei. Sichern Sie die Datei.

b) Führen Sie das Kommando

```
tail -n 20  IrgendeinText.txt | head > IET.txt
```

aus und finden Sie heraus, was dieses Kommando anstellt. Hinweis: Lesen Sie z.B. die Webseite `http://www.selflinux.de/selflinux/html/bash_basic03.html`.

Lösung

In die Datei IET.txt werden die vorletzten 10 Zeilen geschrieben:

- `tail -n 20 IgendeinText.txt` gibt die letzten 20 Zeilen aus
- Die *Pipe* `|` leitet die Ausgabe des letzten Befehls als Eingabe für den nächsten Befehl um
- `head` druckt nun die ersten 10 Zeilen aus
- `>` leitet die Ausgabe eines Befehls in eine Datei um (daher werden die 10 Zeilen nicht ausgedruckt, sondern in die Datei `IET.txt` geschrieben).

c) Verwenden Sie das Kommando `mv`, um die Datei `IrgendeinText.txt` umzubenennen in `MeinText.txt`

Lösung

```
mv IrgendeinText.txt MeinText.txt
```

d) Verwenden Sie das Kommando `cp`, um eine Kopie der Datei `MeinText.txt` namens `NochmalMeinText.txt` zu erstellen

Lösung

```
cp MeinText.txt NochmalMeinText.txt
```

e) Wechseln Sie in Ihr Homeverzeichnis und erstellen Sie eine Kopie des *Verzeichnisses* `vorkurs` namens `vorkurs2`. Welche Option benötigt `cp` hierfür?

Lösung

```
cp -r vorkurs vorkurs2
```

f) Benennen Sie das Verzeichnis `vorkurs2` in `vorkurs-2` um.

Lösung

```
mv vorkurs2 vorkurs-2
```

g) Führen Sie im Homeverzeichnis das Kommando

```
grep -R Informatik *
```

aus. Welche Ausgabe erhalten Sie? Finden Sie anhand der Man Page zu `grep` heraus, was das Kommando macht.

Lösung

grep durchsucht die angegebenen Dateien nach einem Wort. Die Option `-R` bedeutet, dass grep auch rekursiv in Unterverzeichnissen suchen soll. Der `*` steht für eine beliebige Datei, d.h. es wird nach dem Wort `Informatik` in allen Dateien im Homeverzeichnis (und darunter) gesucht.