

Übungsfragen für das Fachgespräch

Christoph Lüth

Praktische Informatik 3, WS 20/21

Zwei Int-Listen sollen *ähnlich* heißen, wenn Sie die gleichen Zahlen unabhängig von ihrer Reihenfolge und Häufigkeit enthalten. Schreiben Sie eine Testfunktion `similar` dafür.

Beispiel:

```
similar [3,2,2,1,3] [1,2,3] ==> True
```

1 [Schwer]

Definieren Sie eine Funktion `protocol`, die wiederholt eine Eingabe von der Konsole liest, in eine Datei "protocol" schreibt, und bei einer Eingabe einer Zeile aus einem Punkt . terminiert.

2 [Schwer]

Implementieren Sie eine **while**-Schleife. Eingabe sind

- ein Startzustand vom Typ `a`,
- eine Testfunktion (`a -> Bool`) und
- eine Zustandsübergangsfunktion (`a -> a`).

Das Ergebnis soll der Endzustand sein.

3 [Schwer]

Gegeben seien Binärbäume:

```
data Tree a = Lf | Br (Tree a) (Tree a)
```

Definieren Sie die Faltung `fold op c t` mit folgender Signatur:

```
fold :: (b -> b -> b) -> b -> Tree a -> b
```

4 [Schwer]

Definieren Sie einen abstrakten Datentyp für Listen, der außer den Elementen zusätzlich ihre Anzahl speichert, so dass ein Zugriff auf die Listenlänge in konstanter Zeit $O(1)$ möglich ist.

5 [Schwer]

Geben Sie einen algebraischen Datentyp für die geometrischen Formen *Kreis* und *Quadrat* an. Definieren Sie eine Funktion, welche die Summe der Flächen einer Liste von Formen berechnet.

6 [Schwer]

Definieren Sie eine Funktion `mean`, die den arithmetischen Durchschnitt einer Liste von ganzen Zahlen berechnet.

Beispiel:

`mean [2,1,5,4,3] ~> 3.0`

7 [Mittel]

Definieren Sie eine Funktion `funny`, die in einer Zeichenkette hinter jedem Vokal den Buchstaben `b` und eine Wiederholung dieses Vokals einfügt.

Beispiel:

`funny "haskell" ~> "habaskebell"`

8 [Mittel]

Definieren Sie eine Funktion `sumOfDigits`, die die Quersumme einer ganzen Zahl zurückliefert.

Beispiel:

`sumOfDigits 178 ~> 16`

9 [Mittel]

Schreiben Sie die Funktion `frequency`, die die Auftrittshäufigkeit von Elementen in einer Liste bestimmt, und als Assoziativliste zurückgibt.

Beispiel:

`frequency "abrakadabra"`
`~> [(5,'a'), (2,'b'), (2,'r'),`
`(1,'k'), (1,'d')]`

10 [Mittel]

Schreiben Sie eine Funktion `runs`, die eine Eingabeliste so in Teillisten zerlegt, dass diese aufsteigend sortiert und maximal lang sind.

Beispiel:

`runs [5,3,7,6] ~> [[5], [3,7], [6]]`

11 [Mittel]

Definieren Sie eine Funktion `binary`, die die Binärrepräsentation einer ganzen Zahl als Zeichenkette berechnet.

Beispiel:

`binary 10 ~> "1010"`

12 [Mittel]

Definieren Sie eine Funktion `read_binary`, die die als Zeichenkette vorliegende Binärrepräsentation in eine ganze Zahl umrechnet.

Beispiel:

```
read_binary "1010" ~> 10
```

13 [Mittel]

Definieren Sie eine Funktion `format`, die eine Zahl in einer Zeichenkette gegebener Länge rechtsbündig ausgibt.

Beispiel:

```
format 4 35 ~> " 35"
```

14 [Leicht]

Definieren Sie eine Funktion `u4x`, die in einer Zeichenkette alle `x` durch ein `u` ersetzt.

Beispiel:

```
u4x "Hexer" ~> "Heuer"
```

15 [Leicht]

Betrachten Sie den vordefinierten Typ

```
data Maybe a = Nothing | Just a
```

Definieren Sie die Funktion `catJusts`, welche aus einer Liste von `Maybe`-Werten die Liste der vorhandenen (mit `Just` konstruierten) Werte zurückgibt.

16 [Leicht]

Definieren Sie eine Funktion `lookup`, die den Wert berechnet, mit dem ein Element in einer Liste assoziiert ist.

Beispiel:

```
lookup 0 [(1,7),(0,4),(3,0)] ~> Just 4
```

17 [Leicht]

Definieren Sie eine Funktion `count`, die berechnet, wie oft ein Element in einer Liste von Listen auftaucht.

Beispiel:

```
count 0 [[0,1],[5],[0,3]] ~> 2
```

18 [Leicht]

Definieren Sie eine Operation (`\`), die alle Auftreten eines Wertes aus einer Liste entfernt.

Beispiel:

```
[0,1,5,0,3] \ 0 ~> [1,5,3]
```

19 [Leicht]

Definieren Sie eine Funktion `zeros`, die zählt, wie oft die Zahl 0 in einer Liste von ganzen Zahlen vorkommt.

Beispiel:

```
zeros [4,0,0,3,2,0] ~> 3
```

20 [Leicht]

Definieren Sie eine Funktion `vowels`, die die Anzahl Vokale in einer Zeichenkette zählt.

Beispiel:

```
vowels "haskell" ~> 2
```

21 [Leicht]