Funktionale Programmierung (ALP 1), WS 2011/12 — 2. Übungsblatt

Test ab 31. Oktober 2011

Bei allen Funktionen ist *generell* eine Typdeklaration anzugeben.

7. Uhrzeiten (10 Punkte)

   Uhrzeiten wie 10:30 Uhr oder 23:59 Uhr sollen als geordnete Paare $(10, 30)$ beziehungsweise (23,59) dargestellt werden.

   Schreiben Sie eine Funktion, die die Dauer zwischen zwei Uhrzeiten in Stunden und Minuten berechnet (negativ, falls die zweite Uhrzeit vor der ersten liegt).

   Sie sollten das Problem in Teilprobleme zerlegen und geeignete Hilfsfunktionen definieren.

8. Uhrzeiten (10 Punkte)

   Im angelsächsischen Raum werden Uhrzeiten in einem 12-Stunden-Bereich angegeben und mit dem Zusatz *a.m.* (Vormittag), *p.m.* (Nachmittag), *noon* oder *midnight* versehen: 0:00 = 12:00 midnight, 0:13 = 12:13 a.m., 10:30 = 10:30 a.m., 12:00 = 12:00 noon, 12:55 = 12:55 p.m., 13:20 = 1:20 p.m., 23:59 = 11:59 p.m.. Die Stundenzahl ist also immer zwischen 1 und 12.

   Schreiben Sie eine Funktion, die dieses Format (als Zeichenkette) berechnet. Wenn die Eingabe keine gültige Uhrzeit darstellt, soll der Text `ungültig` ausgegeben werden.

9. Multiplikationstabelle (10 Punkte)

   Erstellen Sie eine Tabelle für das kleine Einmaleins (die Produkte aller Zahlenpaare zwischen 1 und 10) als Zeichenkette. Die einzelnen Zeilen sind mit `'\n'` abgeschlossen. Das reduzierte Beispiel rechts zeigt, wie so eine Tabelle aussehen könnte, wenn man sie mit `putStr` ausgibt. Die Spalten ihrer Tabelle *müssen* vertikal ausgerichtet sein. (Hinweis: die Funktion `show` wandelt eine Zahl in eine Zeichenkette um.)

```
   |  1  2  10
---+----------
 1 |  1  2  10
 2 |  2  4  20
10 | 10 20 100
```

10. Bedingte Ausdrücke (10 Punkte)

    Schreiben Sie die folgende Funktion als einen *einzigen* (nicht geschachtelten) if-then-else-Ausdruck:

```
test x y z
  | x <= y    = True
  | y <= z    = False
  | otherwise = x < z
```

    Zusatzfrage (0 Punkte): Kommt man auch ganz ohne if-then-else-Ausdruck aus?

11. Perfekte Zahlen (10 Punkte)

    (a) Schreiben Sie eine Funktion, die zu einer Zahl n die Liste ihrer Teiler berechnet.

    (b) Bestimmen Sie jede Zahl zwischen 1 und 1000, bei der die Summe ihrer Teiler größer ist als die Zahl selbst.
    (Hinweis: Die Funktion `sum` berechnet die Summe der Elemente einer Liste)

    (c) Bestimmen Sie alle *perfekten Zahlen* zwischen 1 und 1000: die Zahlen, die gleich der Summe Ihrer Teiler sind.