Vergessen Sie nicht, die unterfertigte eidesstattliche Erklärung mit abzugeben!

185.A03 Funktionale Programmierung WS 2020/2021

Schriftlicher Online-Test 3 auf Papier

Freitag, 28.05.2021, 15:00–17:00 Uhr

Aufgabe	**1**	**2**	**3**	**4**	**5**	**6**	**Summe Punkte**
Punkte	8	18	12	16	18	28	100
Erreichte Punkte							

Aufgabe 1 (4+4 = 10 Punkte)

Gegeben ist der Datentyp `Nat` zur Modellierung natürlicher Zahlen:

```
data Nat = Null | Succ Nat deriving (Eq,Ord)
```

Durch die `deriving`-Klausel in der Typdeklaration von `Nat` wird (u.a.) die Bedeutung der Relatoren `(==)` und `(>)` auf `Nat`-Werten und damit der Relationen *gleich* und *echt größer* auf `Nat`-Werten festgelegt.

Beschreiben Sie die Bedeutung dieser Relationen in der durch die `deriving`-Klausel gegebenen Weise in Worten. Vervollständigen Sie dazu (gdw $\widehat{=}$ genau dann wenn):

1. Zwei `Nat`-Werte m und n sind *gleich* gdw...
2. Ein `Nat`-Wert m ist *echt größer* als ein `Nat`-Wert n gdw ...

Aufgabe 2 (2+2+4+4+(1+1+2+2) = 18 Punkte)

Gegeben ist der Datentyp `Nat` aus Aufgabe 1:

```
data Nat = Null | Succ Nat deriving (Eq,Ord)
```

Implementieren Sie die Addition, Subtraktion, Multiplikation und Quotient und Rest liefernde ganzzahlige Division auf `Nat`-Werten. Alle Funktionen und etwaige Hilfsfunktionen dürfen ausschließlich mit `Nat`-Werten arbeiten und `Nat`-Werte manipulieren. Dabei gilt:

- Ist m kleiner oder gleich n, ist die Differenz von m und n null.
- q und r sind Quotient und Rest der ganzzahligen Division von m und n gdw: $m = q * n + r$ und $0 \leq r < n$. Für $n = 0$ ist die ganzzahlige Division mit Rest nicht definiert.

1. Die Addition:

   ```
   plus :: Nat -> Nat -> Nat
   ```

2. Die Subtraktion:

   ```
   minus :: Nat -> Nat -> Nat
   ```

3. Die Multiplikation:

   ```
   mal :: Nat -> Nat -> Nat
   ```

4. Die Quotient und Rest liefernde ganzz. Division mit Panikfehlerbehandlung bei Nichtdefiniertheit:

   ```
   type Quotient = Nat
   type Rest     = Nat
   data QR       = QR Quotient Rest
   durch :: Nat -> Nat -> QR
   ```

5. Beschreiben Sie knapp, aber gut nachvollziehbar, wie Ihre Implementierungen aus Aufgabe 2.1 bis 2.4 vorgehen.

Aufgabe 3 (6+6 = 12 Punkte)

Lässt sich die Funktion `durch` aus Aufgabe 2.4 mit transparenter nichtverschleiernder Fehlerbehandlung verschieden von Panikfehlerbehandlung implementieren, wenn die Signatur `Nat -> Nat -> QR` von `durch`

1. beibehalten werden muss? Wenn ja, wie? Wen nein, warum nicht?
2. abgeändert werden darf? Wenn ja, wie? Wenn nein, warum nicht?

Aufgabe 4 (10+6 = 16 Punkte)

1. Machen Sie den Typ

   ```
   data Nat = Null | Succ Nat deriving (Eq,Ord)
   ```

   aus Aufgabe 1 zu einer Instanz der Typklasse `Show`.

   Die Ausgabe von `Nat`-Werten soll dabei so erfolgen, wie in den folgenden Beispielen gezeigt:

   ```
   show Null ->> "Null"
   show (Succ Null) ->> "|"
   show (Succ (Succ Null)) ->> "||"
   show (Succ (Succ (Succ (Succ Null)))) ->> "||||"
   show (Succ (Succ (Succ (Succ (Succ Null))))) ->> "++++"
   show (Succ (Succ (Succ (Succ (Succ (Succ Null)))))) ->> "++++ |"
   show (Succ (Succ (Succ (Succ (Succ (Succ (Succ (Succ (Succ Null))))))))) ->> "++++ ||||"
   show (Succ (Succ (Succ (Succ (Succ (Succ (Succ (Succ (Succ (Succ Null)))))))))) ->> "++++ ++++"
   show (Succ (Succ (Succ (Succ (Succ (Succ (Succ (Succ (Succ (Succ (Succ Null)))))))))))
                                                                                    ->> "++++ ++++ |"
   und im gleichen Stil so weiter.
   ```

   Vollständige Gruppen von je fünf Strichen `"|||||"` sollen also durch Folgen von je vier `'+'`-Zeichen `"++++"` dargestellt werden in Anlehnung an die handschriftliche Praxis, wo man zur einfacheren Lesbarkeit statt 5er-Strichgruppen meist durchstrichene 4er-Strichgruppen ~~||||~~ schreibt.

   Weiters soll gelten:

   - Verschiedene Vorkommen von 5er-Gruppendarstellungen sind in der Ausgabezeichenreihe durch ein Leerzeichen getrennt, ebenso 5er-Gruppendarstellungen und Strichgruppen.
   - Verschiedene Striche in einer Strichgruppe folgen unmittelbar aufeinander und sind nicht durch Leerzeichen getrennt.
   - Am Ende der Ausgabezeichenreihe steht kein Leerzeichen; gleich, ob die Ausgabezeichenreihe mit einer 5er-Gruppendarstellung oder einem Strich endet.

2. Erklären Sie knapp, aber gut nachvollziehbar, wie Ihre Instanzbildung vorgeht und warum sie die Anforderungen erfüllt.

Aufgabe 5 (6+6+6 = 18 Punkte)

$$f : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$$

$$f(z, z') =_{df} \sum_{i=0}^{|z|} i^{|z'|}$$

Implementieren Sie die Haskell-Rechenvorschrift

```
f :: (Integer,Integer) -> Integer
```

zur Berechnung von f auf folgende unterschiedliche Weisen:

1. Direkt rekursiv oder abgestützt auf eine direkt rekursive (Hilfs-) Funktion.
2. Mithilfe von Listenkomprehension(en) und der in Haskell vordefinierten Funktion zum Summieren der Elemente einer Liste numerischer Werte.
3. Mithilfe von Listenkomprehension(en) und in Haskell vordefinierter Funktionale zum Transformieren und Falten von Listen.

Aufgabe 6 ((2+2)+2+(1+1)+8+(5+5)+2 = 28 Punkte)

Gegeben ist folgende Implementierungszeile:

```
k f x y z = f (x,y,z)
```

1. Geben Sie den allgemeinsten Typ von `k`
   (a) vollständig, aber nicht überflüssig geklammert an.
   (b) gemäß Klammereinsparungsregeln von Haskell minimal geklammert an.
2. Ist der allgemeinste Typ von `k` monomorph oder polymorph? Wenn polymorph, geben Sie die Polymorphieart möglichst genau an, woran man sie erkennt und welche synonymen Bezeichnungen es dafür gibt.
3. Klammern Sie die (i) rechte und (ii) linke Seite der Implementierungszeile von `k` vollständig, aber nicht überflüssig.
4. Zeigen Sie, dass der rechtsseitige Funktionsterm in der Implementierungszeile den in der Signaturzeile geforderten Typ besitzt. Geben Sie dazu den Typ aller Teilterme der links- und rechtsseitigen Funktionsterme nach dem Muster des folgenden Beispiels an:

```
        "Hallo "   ++   "Welt!"
       :: String        :: String
   :: String -> String -> String
            :: String
```

5. Werten Sie den Ausdruck:

   ```
   k (\(x,y,z) -> x*y+z) (2+3) (3*5) (7-5)
   ```

   Schritt für Schritt
   (a) linksnormal
   (b) linksapplikativ

   aus. Ein Schritt ist dabei <u>ein</u> Expansionsschritt oder <u>eine</u> Operatoranwendung eines Simplifikationsschritts.
6. Was ist die Bedeutung von `k`, was macht es? Zu welcher Funktion ist `k` ähnlich? Warum?

Vergessen Sie nicht, die unterfertigte eidesstattliche Erklärung mit abzugeben!