8. Übung | **Informatik A** | **WS 13/14**

Klaus Kriegel — **Abgabe: 16.12.2013, 12:00 Uhr**

Aufgabe 1: **Rekursion** (4 Punkte)

Nutzen Sie das Beispiel aus der Vorlesung zur schnellen Berechnung der Fibonacci-Zahlen mit einem primitiv rekursiven Ansatz, um eine Haskell-Implementierung der Funktion

$$g : \mathbb{N} \longrightarrow \mathbb{N} \qquad \text{mit} \quad g(n) = \begin{cases} n & \text{falls } n \leq 2 \\ g(n-1) + 2g(n-2) - g(n-3) & \text{sonst} \end{cases}$$

zu realisieren, so dass die Ausführung nur lineare Zeit erfordert.

Aufgabe 2: **Tupel** $(2 + 2 + 2 + 2$ Punkte)

Wir führen Datentypen `type Point = (Float,Float)` und `type Line = (Float,Float)` ein, wobei die durch `(a,b)` repräsentierte Gerade die Gleichung $y = ax + b$ erfüllen soll. Implementieren Sie die folgenden Funktionen:

a) `pointOnLine, pointOverLine :: Point -> Line -> Bool` geben `True` zurück, wenn der Punkt auf der Geraden bzw. darüber liegt.

b) `lineThrough :: Point -> Point -> Line` berechnet die Gerade durch zwei Punkte und gibt Fehlermeldungen, wenn die Eingabepunkte identisch sind oder gleiche x-Koordinaten haben.

c) `crossing :: Line -> Line -> Point` berechnet den Schnittpunkt der Geraden (sofern existent) und gibt Fehlermeldungen bei identischen oder parallelen Geraden.

d) `parallelThrough :: Line -> Point -> Line` berechnet die Gerade, welche parallel zur gegebenen Gerade durch den gegebenen Punkt verläuft.

Aufgabe 3: **Binomialkoeffizienten** $(2 + 2 + 1 + 2 + 3$ Punkte)

Der Binomialkoeffizient $\binom{n}{k}$ ist für alle Paare von natürlichen Zahlen mit $n \geq k$ als der folgende Quotient definiert:

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!} = \frac{n \cdot (n-1) \cdot \ldots \cdot (n-k+1)}{k!} \qquad \text{insbesondere} \qquad \binom{0}{0} = 1$$

a) Implementieren Sie eine Funktion `binomDef :: Integer -> Integer -> Integer` zur Berechnung des Binomialkoeffizienten nach Definition.

b) Für alle $n > k > 0$ gilt die Rekursion $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$. Beweisen Sie diese Rekursion anhand der Definition.

c) Implementieren Sie eine Funktion `binomRec :: Integer -> Integer -> Integer` zur Berechnung des Binomialkoeffizienten mit dieser Rekursion, wobei man die Verankerungen $\binom{n}{n} = \binom{n}{0} = 1$ verwendet.

d) Implementieren Sie eine Funktion `countCalls :: Integer -> Integer -> Integer`, welche die Anzahl der Funktionsaufrufe von `binomRec` bei der Berechnung eines Binomailkoeffizienten zählt. Überzeugen Sie sich an Beispielen, dass diese Anzahl bei der Berechnung von $\binom{n}{\lfloor n/2 \rfloor}$ exponentiell groß wird, d.h. $\geq c \cdot 2^{dn}$ für geeignete Konstanten $c, d \in \mathbb{R}^+$ wird. Stellen Sie eine Vermutung auf, wie man die Konstanten geeignet wählen kann.

e) Beweisen Sie mit vollständiger Induktion, dass die Laufzeit der rekursiven Funktion (gemessen in Aufrufen von `binomRec`) für die Berechnung von $\binom{n}{\lfloor n/2 \rfloor}$ mindestens exponentiell ist.

Hinweise: In Teil e) muss man die Behauptung konkret mit Konstanten c, d formulieren. Die Idee dafür sollte aus Teil d) kommen. Wer Probleme beim Finden der Konstanten hat, kann sich zuerst auf die geraden Fälle konzentrieren und die Werte von `countCalls` für die Eingabewerte $2k$ und k mit der Funktion 2^k vergleichen. Danach muss man nur noch $2k$ durch n substituieren. Es kommt nicht darauf an, möglicht große Werte für c und d zu finden, sondern möglicht einfache, mit denen sich Teil e) leicht beweisen lässt.