Functioneel programmeren

Practicum 1: *Getallen*

- Tijdens dit eerste practicum gaan we op uiteenlopende manieren aan de slag met getallen. Het practicum bestaat uit tien opgaven. Allereerst definiëren we een aantal functies die van doen hebben met getallensystemen gebaseerd op verschillende grondtallen (opgave 1 t/m 7). Daarna schrijven we functies om Gray-coderingen (opgave 8), look-and-say-reeksen (opgave 9) en Keith-getallen (opgave 10) te produceren.
- Ingeleverde code wordt niet alleen beoordeeld op basis van correctheid, maar ook op basis van programmeerstijl. Zorg er daarom voor dat je je code van voldoende en duidelijk commentaar voorziet. Het gebruik van standaardfuncties (met name hogere-ordefuncties als *map* en *filter*) en lijstcomprehensies wordt aangemoedigd. Uiteraard mag je zoveel hulpfuncties definiëren als je nodig acht.
- Werk in teams van maximaal twee personen. Lever je code, in een enkel Haskell-bronbestand, in met behulp van het Submit-systeem (`http://www.cs.uu.nl/docs/submit/`). Maak, door middel van commentaar boven in je bronbestand, duidelijk wie de leden van het team zijn en welke compiler gebruikt is. Alleen compileerbare programma's komen in aanmerking om beoordeeld te worden! Toegestane compilers zijn: Helium (versie 1.6 of hoger) en GHC (versie 6.8.2 of hoger).
- De deadline voor inleveren is **maandag 2 maart 2009** om **23:59** uur.

Getallensystemen

In het dagelijks leven zijn we gewoon om binnen het *decimale* (dat wil zeggen: tientallige) getallensysteem te rekenen. Getallen binnen dit systeem worden samengesteld uit tien cijfers: `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8` en `9`. Een reeks cijfers wordt dan geïnterpreteerd als een som van 10-machten. Bijvoorbeeld: de notatie `423` wordt gebruikt om het getal $4 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 = 423$ aan te duiden.

Opgave 1. Schrijf een functie

$$\mathit{fromDec} :: [\mathit{Int}] \to \mathit{Int}$$

die voor een reeks decimale cijfers de bijbehorende interpretatie als som van 10-machten produceert. Bijvoorbeeld: $\mathit{fromDec}\ [4, 2, 3]$ geeft 423. ∎

Opgave 2. Schrijf een functie

$$\mathit{toDec} :: \mathit{Int} \to [\mathit{Int}]$$

die een getal omzet naar de bijbehorende decimale cijferreeks. Bijvoorbeeld: $\mathit{toDec}\ 423$ geeft $[4, 2, 3]$. ∎

Het decimale getallensysteem is niet het enige systeem dat we kunnen gebruiken om mee te rekenen. Geheugencellen van computers kunnen bijvoorbeeld maar twee waarden aannemen; in de context van een computer is het dus heel natuurlijk om binnen het *binaire* (tweetallige) systeem te rekenen. De cijfers binnen dat systeem zijn `0` en `1` en cijferreeksen worden geïnterpreteerd als sommen van 2-machten. Bijvoorbeeld: de reeks `100110` is een binaire representatie van het getal $1{\cdot}2^5{+}0{\cdot}2^4{+}0{\cdot}2^3{+}1{\cdot}2^2{+}1{\cdot}2^1{+}0{\cdot}2^0 = 1{\cdot}32{+}0{\cdot}16{+}0{\cdot}8{+}1{\cdot}4{+}1{\cdot}2{+}0{\cdot}1 = 38$.

Opgave 3. Schrijf een functie

$$\mathit{fromBin} :: [\mathit{Int}] \to \mathit{Int}$$

die voor een reeks binaire cijfers de bijbehorende interpretatie als som van 2-machten produceert. Bijvoorbeeld: $\mathit{fromBin}\ [1,0,0,1,1,0]$ geeft 38. ∎

Opgave 4. Schrijf een functie

$$\mathit{toBin} :: \mathit{Int} \to [\mathit{Int}]$$

die een getal omzet naar de bijbehorende binaire cijferreeks. Bijvoorbeeld:

$$\mathit{toBin}\ 38$$

geeft $[1,0,0,1,1,0]$. ∎

Een ander voorbeeld van een niet-decimaal getallensysteem is het *hexadecimale* systeem, dat gebruik maakt van maar liefst zestien cijfers: `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `a`, `b`, `c`, `d`, `e` en `f`. Binnen het systeem staan de cijfers `a`, `b`, `c`, `d`, `e` en `f` voor de getallen 10, 11, 12, 13, 14 en 15. Cijferreeksen worden geïnterpreteerd als sommen van 16-machten. Dus: $\texttt{f4c8} = 15 \cdot 16^3 + 4 \cdot 16^2 + 12 \cdot 16^1 + 8 \cdot 16^0 = 15 \cdot 4096 + 4 \cdot 256 + 12 \cdot 16 + 8 \cdot 1 = 62664$.

In het algemene geval maakt een getallensysteem gebaseerd op een grondtal n dus gebruik van n cijfers en worden cijferreeksen geïnterpreteerd als n-machten. Voor de eerste 10 cijfers in een systeem gebruiken we dan de symbolen `0` tot en met `9`. Zijn er meer cijfers nodig (als n groter is dan 10), dan gebruiken we de letterstekens van *a* tot `z`. In wat volgt zullen we ons beperken tot keuzes voor n tussen 2 en 36, zodat we van de genoemde symbolen er ook daadwerkelijk genoeg hebben om alle getallen uit te drukken.

Opgave 5. Schrijf een functie

$$\mathit{fromBase} :: \mathit{Int} \to [\mathit{Char}] \to \mathit{Int}$$

die voor een grondtal tussen 2 en 36 (inclusief) en een cijferreeks de bijbehorende interpretatie produceert. Bijvoorbeeld:

- *fromBase* 10 `"423"` geeft 423;
- *fromBase* 2 `"100110"` geeft 38;
- *fromBase* 16 `"f4c8"` geeft 62664;
- *fromBase* 23 `"5j"` geeft 134. ∎

Opgave 6. Schrijf een functie

$$toBase :: Int \to Int \to [Char]$$

die voor een grondtal tussen 2 en 36 (inclusief) en een getal de bijbehorende cijferreeks produceert. Bijvoorbeeld:

- *toBase* 10 423 geeft `"423"`;
- *toBase* 2 38 geeft `"100110"`;
- *toBase* 16 62664 geeft `"f4c8"`;
- *toBase* 23 134 geeft `"5j"`. ∎

Opgave 7. Schrijf een functie

$$numbers :: Int \to [String] \to [(String, Int)]$$

die voor een grondtal tussen 2 en 36 (inclusief) en een lijst met woorden de woorden produceert die overeenkomen met een getal. Bijvoorbeeld:

$$numbers\ 12\ [\texttt{"ace"}, \texttt{"face"}, \texttt{"faces"}]$$

geeft $[(\texttt{"ace"}, 2766), (\texttt{"face"}, 64206)]$. ∎

Gray-coderingen

Een *Gray-codering* is een codering van de natuurlijke getallen in in cijferreeksen waarbij de cijferreeksen voor twee opeenvolgende getallen altijd op slechts één positie verschillen. Een voorbeeld van een binaire Gray-codering (met de cijfers `0` en `1`) is:

$$\begin{array}{rcr} 0 & \mapsto & \texttt{0} \\ 1 & \mapsto & \texttt{1} \\ 2 & \mapsto & \texttt{11} \\ 3 & \mapsto & \texttt{10} \\ 4 & \mapsto & \texttt{110} \\ 5 & \mapsto & \texttt{111} \\ 6 & \mapsto & \texttt{101} \\ 7 & \mapsto & \texttt{100} \\ & \vdots & \end{array}$$

Opgave 8. Schrijf een (efficiënte) functie

$$grayCode :: Int \to ([Char] \to Int, Int \to [Char])$$

die voor een gegeven aantal cijfers tussen 2 en 36 (inclusief) een Gray-codering construeert, dat is, een tweetal functies waarvan de eerste een cijferreeks interpreteert als een getal en de tweede voor een getal de bijbehorende cijferreeks produceert. ∎

Look-and-say-reeksen

Conways *look-and-say-reeks* is een reeks getallen waarin elk element van de reeks een 'beschrijving' van het vorige element vormt. Bijvoorbeeld, als we de reeks beginnen met 1, dan:

- is het tweede element 11 (want het voorgaande element bestond uit eenmaal een één);
- is het derde element 21 (want het voorgaande element bestond uit tweemaal een één);
- is het vierde element 1211 (want het voorgaande element bestond uit eemaal een twee en eenmaal een één);
- enzovoort.

Opgave 9. Schrijf een functie

$$\mathit{lookAndSay} :: \mathit{Int} \to [\mathit{String}]$$

die gegeven een eerste element de bijbehorende (oneindige) look-and-say-reeks produceert. Bijvoorbeeld: *take* 6 (*lookAndSay* 1) geeft

$$[\texttt{"1"}, \texttt{"11"}, \texttt{"21"}, \texttt{"1211"}, \texttt{"111221"}, \texttt{"312211"}].$$

■

Keith-getallen

Tenslotte bekijken we de volgende manier om tot een oneindige reeks van getallen te komen. Elke reeks is gebaseerd op de decimale cijferreeks voor een bepaald natuurlijk getal en de lengte n van die cijferreeks. Bijvoorbeeld: voor het getal 123 bestaat de cijferreeks uit de cijfers `1`, `2` en `3` en hebben we verder dat $n = 3$ (want de lengte van de cijferreeks is 3). De eerste n elementen van de reeks zijn dan de cijfers van de cijferreeks en elk volgend element wordt bepaald door de n voorafgaande elementen op te tellen. Bijvoorbeeld:

- het getal 14 genereert de reeks 1, 4, 5, 9, 14, 23, 37, 60, $\cdots$;
- het getal 123 genereert de reeks 1, 2, 3, 6, 11, 20, 37, 68, $\cdots$;
- het getal 197 genereert de reeks 1, 9, 7, 17, 33, 57, 107, 197, $\cdots$.

Een natuurlijk getal is een *Keith-getal* als het groter is dan 9 en zelf voorkomt in de reeks die het op bovenstaande manier genereert. Dus: 14 en 197 zijn Keith-getallen (want ze komen zelf in de door hen genereerde reeks voor), maar 123 is dat niet (want 123 komt niet in zijn eigen reeks voor).

Opgave 10. Construeer een (oneindige) lijst

$$\mathit{keithGetallen} :: [\mathit{Int}]$$

die alle Keith-getallen in oplopende volgorde bevat. Bijvoorbeeld:

$$\mathit{take}\ 5\ \mathit{keithGetallen}$$

geeft $[14, 19, 28, 47, 61]$. (Als je gebruik maakt van de GHC, dan mag je de lijst ook het type $[\mathit{Integer}]$ geven; zoals je zult merken zijn er maar weinig Keith-getallen zijn die in het type Int passen.) ■