Functioneel Programmeren – opgaven practiucm 3

Voorbereiding. Run in een terminal window:

cabal update

cabal install twentefp-rosetree

Neem in uw programma's op:

import RoseTree

Download van Blackboard|CourseMaterials de file `StandaardWebpag.zip`, en open de file `Standaard Webpagina.html` in een browser[1].

U hebt nu het volgende type tot uw bschikking:

```
data  RoseTree  =  RoseNode String [RoseTree]
```

Bomen van het type `RoseTree` hebben bij elke knoop een string (mogelijk leeg) en een lijst van subbomen (als deze lijst leeg is, is de betreffende knoop een blad).

Bovendien zijn er de functies:

```
showTree
showTreeList
```

die bomen van het type `RoseTree` grafisch kunnen weergeven. De functie `showTree` werkt op een enkele boom van type `RoseTree`, de functie `showTreeList` op een lijst van bomen van dat type.

Opgave 1. Om bomen van een ander type dan `RoseTree` grafisch weer te kunnen geven, moeten ze eerst omgezet worden naar dat type. In deze opgave moet u bij elk onderdeel zelf een voorbeeldboom van het betreffende type definiëren en die vervolgens grafisch vertonen.

a. Gegeven is het type

```
data Tree1a  =  Leaf1a Number  |  Node1a Number Tree1a Tree1a,
```

dus binaire bomen met integers aan de bladeren en aan de knopen. Definieer een funktie `pp1a` (voor pre-processor) die een boom van type `Tree1a` omzet in een boom van type `RoseTree`.

[1] Er zijn verschillende browsers getest: *Chrome*, *Opera*, en *Internet Explorer* zijn OK, *Firefox* en *Safari* zijn mogelijk niet helemaal stabiel.

b. Definieer een type `Tree1b` voor binaire bomen die getallenparen (type: `(Number,Number)`) aan knopen en bladeren bevatten. Definieer een funktie `pp1b` die bomen van type `Tree1b` omzet in bomen van type `RoseTree`.

c. Idem, maar nu voor een type `Tree1c` dat alleen integers aan de knopen bevat, en geen informatie aan de bladeren (hint: voeg de lege string toe aan de bomen van type `RoseTree`).

d. Tenslotte, definieer het type `Tree1d` dat geen informatie aan de knopen bevat, en getallenparen aan de bladeren. Het type moet op elke knoop een willekeurig aantal subbomen kunnen hebben.

Opgave 2.

a. Definieer een functie die bij elk getal in een boom van type `Tree1a` een getal `x` optelt. Laat enkele voorbeelden grafisch zien.

b. Definieer een functie die elk getal in een boom kwadrateert. Laat het resultaat grafisch zien.

c. Definieer een funktie

```
mapTree :: (Number -> Number) -> Tree1a -> Tree1a
```

die een funktie `f` (type `Number -> Number`) toepast op alle getallen in een boom van type `Tree1a`.

Definieer de functies van onderdeel **a** en **b** met behulp van `mapTree` (en laat het resultaat in één plaatje grafisch zien).

d. Definieer een functie `telopNode` die in een boom van type `Tree1b` alle getallenparen vervangt door de som van beide getallen. Laat het resultaat grafisch zien.

e. Definieer een variant van `mapTree` waarmee elke functie

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

op alle paren in een boom van type `Tree1b` kan worden toegepast. Demonstreer uw functie met optellen, aftrekken, vermenigvuldigen (maak gebruik van lambda-abstractie).

Opgave 3.

a. Definieer een funktie `binspiegel` die een binaire boom van type `Tree1a` gespiegeld weergeeft. Test de functie door de voorbeeldexpressie van opgave **??**a samen met zijn gespiegelde vorm grafisch te tonen. Check dat twee keer spiegelen de oorspronkelijke boom weer oplevert.

b. Schrijf een variant op deze spiegelfunctie die werkt voor bomen van type `Tree1d`, en wel zodanig dat ook de getallenparen aan de bladeren omgedraaid worden.

Definitie. Een binaire boom met getallen is *gesorteerd* als voor élke knoop in die boom geldt dat elk getal in de *linker* subboom onder die knoop kleiner dan, of gelijk is aan het getal aan die knoop, en elk getal in de *rechter* subboom groter.

Opgave 4. Ga uit van type `Tree1c`, dus alléén getallen aan de interne knopen, en níet aan de bladeren. Type `Tree1a` zou bij onderstaande deelopdrachten het probleem opleveren dat een boom dan alleen een oneven aantal getallen kan bevatten.

a. Schrijf een funktie *t_insert* die een getal toevoegt aan een *gesorteerde* boom van type `Tree1c`. Het is het handigst om het getal toe te voegen aan een blad van de boom.

b. Schrijf een funktie `makeTree` die een gesorteerde boom van type `Tree1c` maakt van een (ongesorteerde) lijst van getallen. Gebruik de funktie `t_insert`. Definieer deze functie op *twee* manieren: recursief, en met een `fold`-functie.

c. Schrijf een funktie `makeList` die uit een boom weer een lijst maakt. Indien de boom gesorteerd is, moet die sortering gehandhaafd blijven.

d. Combineer eerdere functies uit deze opgave tot een functie die lijsten sorteert door van een lijst eerst een gesorteerde boom te maken, en vervolgens de boom weer om te zetten naar een lijst.

e. Het omgekeerde van **d**: combineer de functies tot een functie die bomen van type `Tree1c` sorteert.

Opgave 5. Schrijf een funktie `zoek` die, gegeven een getal `n` en een gesorteerde boom `t` van type `Tree1c`, díe subboom van `t` oplevert waarvan de wortel (engels: root) gelijk is aan het getal `n`. Als het getal `n` niet in de boom voorkomt, moet de funktie een foutmelding geven.

Opgave 6. Definieer een funktie `totDiepte` die, gegeven een getal `m` en gegeven een boom `t` van type `Tree1a`, elke tak in boom `t` afsnijdt op lengte `m` (tenminste, als die tak langer is dan `m`). Knopen in de oorspronkelijke boom kunnen dus bladeren zijn in de resulterende boom.
Hint: deze opgave bevat een *dubbele* recursie: zowel over getallen als over de struktuur van de boom.

Opgave 7. Een *pad* in een binaire boom is een string die bestaat uit de letters `'l'` en `'r'`, respectievelijk voor *links* en *rechts*. Begin bij de wortel van de boom, en ga naar de linker (resp. rechter) subboom als de eerstvolgende letter in het pad een `'l'` (resp, `'r'`) is. Een pad "wijst" dus naar de wortel van een subboom.

Ga bij deze opgave uit van bomen van type `Tree1a`.

a. Schrijf een functie `vervang` die in een gegeven een boom `t` het getal bij een knoop (aangewezen door een pad `p`) vervangt door een gegeven getal `n`.

b. Schrijf een funktie `subboom` die, gegeven een pad en gegeven een boom, de subboom oplevert die door het pad wordt aangewezen. Als het pad te lang is, moet de funktie een `error` opleveren.

c – toegift. Een blad is *buur* van een ander blad als er geen andere bladen tussen zitten. Een blad in een boom kan worden aangeduid door een pad naar dat blad. Schrijf twee functies `linkerbuur` en `rechterbuur` die gegeven een boom, en gegeven een blad in die boom (aangeduid door een pad) de linker, resp. de rechter buur van dat blad opleveren (eveneens in de vorm van het pad). Bij het meest linker en meest rechter blad moet zo nodig een foutmelding worden gegeven.

Maak het resultaat van uw functies grafisch zichtbaar door `-1` bij de buurbladeren van een gegeven blad te zetten.

Opgave 8. Een binaire boom is *gebalanceerd* als alle takken zoveel mogelijk even lang zijn (dus hoogstens één stap in lengte verschillen). Gebruik bomen van type `Tree1c`.

a. Schrijf een functie die *test* of een boom gebalanceerd is.

b. Schrijf een functie die van een boom een gebalanceerde boom *maakt* (hint: gebruik type `Tree1c`, en werk via de tussenstap van lijsten. Ga na waarom deze benadering voor bomen van type `Tree1c` makkelijker is dan voor bomen van type `Tree1a`).

Ga met uw testfunctie uit onderdeel **a** na of uw functie uit onderdeel **b** goed werkt.