Informatica Instituut, Faculteit Wiskunde en Informatica, UU.
In elektronische vorm beschikbaar gemaakt door de TBC van A−Eskwadraat.
Het college INFOFP werd in 2003/2004 gegeven door Johan Jeuring.
De uitwerking is samengesteld/gemaakt door
Diederik Royers ∧ Roland Vaandrager ∧ Wouter Duivesteijn.

Uitwerking[1] Functioneel Programmeren (INFOFP) 17 maart 2004

Opgave 1

Wat is de waarde van de onderstaande expressie?
`[ y-x | x <- [1,2], y <- [3..5]]` *(5 punten)*

a) `[-2,-3,-4,-1,-2,-3]`

b) `[2,3,4,1,2,3]`

c) `[[-2,-1,-3],[-2,-4,-3]]`

d) `[[2,1],[3,2],[4,3]]`

Antwoord:
Antwoord b is correct. Allereerst wordt voor x de waarde 1 gekozen en voor y achtereenvolgens de waarden 3, 4 en 5. Daarna wordt voor x 2 gekozen, en voor y weer 3, 4 en 5. Met de gekozen waarden wordt $y - x$ uitgevoerd, en de resultaten worden in één lijst gezet.

Opgave 2

Wat is de waarde van de onderstaande expressie?
`sum (takeWhile (<100) [10,50..])` *(5 punten)*

a) 60

b) 160

c) 150

d) geen antwoord, oneindige lus

Antwoord:
Antwoord c is correct. De lijst `[10,50..]` begint bij 10 en gaat met stappen van 40 omhoog. Dus achtereenvolgens $10, 50, 90, 130, \ldots$. Deze lijst wordt afgekapt zodra een waarde ≥ 100 is. Het resultaat is dus $10 + 50 + 90 = 150$.

Opgave 3

Gegeven is het datatype:

```
data MyList a = Cons (MyList a) a | Nil
```

Wat is het type van `Cons 'a' Nil`? *(5 punten)*

a) `MyList`

[1] Deze uitwerkingen zijn met de grootste zorg gemaakt. In geval van fouten kan de TBC niet verantwoordelijk worden gesteld, maar wordt zij wel graag op de hoogte gesteld: `tbc@a-eskwadraat.nl`

b) `MyList Char`

c) `MyList a -> a -> MyList a`

d) typeringsfout

Antwoord:
Antwoord d is correct. Het type van `'a'` is `Char` en het type van `Nil` is `MyList a`. In de gegeven expressie staan de twee argumenten in de verkeerde volgorde.

Opgave 4

Schrijf het meest algemene type op van de onderstaande expressie. Een afleiding is niet nodig. Als de expressie een typeringsfout bevat, schrijf dan op: 'typeringsfout'.

`foldr (++)` *(10 punten)*

Antwoord:
Het type van `foldr` is `(a -> a -> b) -> a -> [a] -> b`. Het type van `(++)` is `[a] -> [a] -> [a]`. Het type van `foldr (++)` wordt als volgt gevormd: `(a -> a -> b)` kan worden weggelaten, omdat `(++)` de rol daarvan speelt. In de rest van het type van `foldr` moeten we steeds zowel `a` als `b` vervangen door `[a]`. Het type van `foldr (++)` is dus `[a] -> [[a]] -> [a]`.

Opgave 5

Gegeven is het volgende datatype:

```
data Boom a = Tak (Boom a) (Boom a) | Blad a
```

Wat is het type van de constructorfunctie `Tak`? *(5 punten)*

Antwoord:
Een `Tak` heeft als argumenten twee keer `Boom a`, en levert een `Boom a` op. Dus is het type van `Tak` gelijk aan `Boom a -> Boom a -> Boom a`.

Opgave 6

Schrijf een functie die telt hoeveel klinkers (a, e, i, o en u) er voorkomen in een `Boom` die letters (`Char`) bevat. Noem de functie `telKlinkers` en schrijf ook het type op. *(15 punten)*

Antwoord:
Een mogelijke functie is:

```
telKlinkers :: (Boom Char) -> Int
telKlinkers (Blad 'a') = 1
telKlinkers (Blad 'e') = 1
telKlinkers (Blad 'i') = 1
telKlinkers (Blad 'o') = 1
telKlinkers (Blad 'u') = 1
telKlinkers (Blad _)   = 0
telKlinkers (Tak a b)  = telKlinkers a + telKlinkers b
```

Opgave 7

Schrijf een functie die bepaalt of alle elementen van een `Boom` voldoen aan een gegeven predicaat. Noem deze functie `allBoom` en schrijf ook het meest algemene type op. Het predicaat is een functie die gegeven een element een `Bool` oplevert. Voorbeeld van gebruik:
`allBoom even (Tak (Blad 2) (Tak (Blad 3) (Blad 4)))` geeft `false`. *(15 punten)*

Antwoord:
Een mogelijke functie is:

```
allBoom :: (a -> Bool) -> (Boom a) -> Bool
allBoom p (Blad a)  = p a
allBoom p (Tak a b) = allBoom p a && allBoom p b
```

Opgave 8

Breid het datatype `Boom` uit, zodanig dat ook lege bomen gerepresenteerd kunnen worden.
(10 punten)

Antwoord:
Maak er een extra constructor `Leeg` bij. De definitie wordt dan:

```
data Boom a = Tak (Boom a) (Boom a)
            | Blad a
            | Leeg
```

Opgave 9

Schrijf een functie `filterBoom`, die, gegeven een predicaat en een `Boom`, alle bladeren die niet voldoen aan het predicaat vervangt door de lege `Boom`. Takken en bladeren die wel aan het predicaat voldoen worden ongemoeid gelaten. Schrijf ook het meest algemene type op.
(15 punten)

Antwoord:
Een mogelijke functie is:

```
filterBoom :: (a -> Bool) -> (Boom a) -> (Boom a)
filterBoom _ Leeg      = Leeg
filterBoom p (Blad a)  | p a       = Blad a
                       | otherwise = Leeg
filterBoom p (Tak a b) = Tak (filterBoom p a) (filterBoom p b)
```

Opgave 10

Schrijf een functie die **alle** lege bomen verwijdert uit een niet-lege boom:

```
verwijderLeeg ::  Boom a -> Boom a
```

Hierbij nemen we aan dat voor alle bomen geldt dat een `Tak l r` met een lege deelboom (`l` of `r`) gelijk is aan de andere deelboom (`r` of `l`). *(15 punten)*

Antwoord:
Een mogelijke functie is:

```
verwijderLeeg           :: Boom a -> Boom a
verwijderLeeg (Tak l r) =  tak (verwijderLeeg l) (verwijderLeeg r)
verwijderLeeg boom      =  boom

tak        :: Boom a -> Boom a -> Boom a
tak Leeg r =  r
tak l Leeg =  l
tak l r    =  Tak l r
```