Informatica Instituut, Faculteit Wiskunde en Informatica, UU.
In elektronische vorm beschikbaar gemaakt door de TEX van A−Eskwadraat.
Het college INFOFP werd in 2003/2004 gegeven door Johan Jeuring.

Functioneel Programmeren (INFOFP)
6 april 2004

Opgave 1 *(7 punten)*

Geef van de volgende expressie aan wat haar type is. *Vergeet overloading* en neem aan dat getallen simpelweg van het type `Int` zijn.

```
map div [1, 2, 3]
```

a) `[Int -> Int]`

b) `[Int]`

c) `[Int] -> [Int]`

d) typefout

Opgave 2 *(7 punten)*

Geef aan welke van de definities **niet** equivalent is aan deze definitie:

```
all p xs = and (map p xs)
```

a) `all p = and .  map p`

b) `all p xs = (and .  map p) xs`

c) `all p = and (map p)`

d) `all = \p xs -> and (map p xs)`

Opgave 3 *(7 punten)*

Wat is de waarde van de onderstaande expressie?

```
map ((/= 0) .  (`mod` 3)) [12, 10, 72, 4]
```

a) `[12, 72]`

b) `[False, True, False, True]`

c) `[True, False, True, False]`

d) `[10, 4]`

Opgave 4 *(7 punten)*

Gegeven zijn de types van `sequence` en `putChar`:

```
sequence ::  [IO a] -> IO [a]
putChar  ::  Char -> IO ()
```

Wat is het type van onderstaande expressie?

```
sequence (map putChar "De zon schijnt")
```

a) `IO [a]`

b) `IO [()]`

c) `[IO ()]`

d) `[Char] -> [IO ()]`

Opgave 5 *(15 punten)*

Schrijf een functie die regels inleest (gebruik makend van `getLine ::  IO String`) totdat de gebruiker een lege regel invoert. De functie heeft een IO type en levert de ingelezen regels inclusief de lege op: `getLines ::  IO [String]`

Opgave 6 *(15 punten)*

Schrijf een functie die de index van het *laatste* voorkomen van een element in een lijst oplevert. Indices beginnen bij nul. Bijvoorbeeld: `lastIndexOf 3 [1,2,3,5,3,2,1]` geeft 4. Je mag aannemen dat het element voorkomt in de lijst. Schrijf een zo algemeen mogelijk type op bij deze functie.

Bonus: Als het je lukt om deze functie helemaal met standaardfuncties (en dus zonder recursie) te schrijven, dan krijg je 5 bonuspunten. Daarmee kun je in principe boven de 10 uitkomen voor deze toets. Het eindcijfer van het vak kan echter niet meer dan een 10 zijn.

Opgave 7 *(10 punten)*

Gegeven is het datatype `Boom` en de functie `foldBoom`:

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

foldBoom :: (a -> b) -> (b -> b -> b) -> Boom a -> b

foldBoom blad tak (Blad x) = blad x

foldBoom blad tak (Tak links rechts) =
       tak (foldBoom blad tak links) (foldBoom blad tak rechts)
```

Schrijf met behulp van `foldBoom` een functie die het aantal bladeren in een `Boom` bepaalt:

```
aantal ::  Boom a -> Int
```

Opgave 8 *(8 punten)*

Gegeven is het datatype om breuken mee te representeren:

```
data Getal = Breuk Int -- teller
                   Int -- noemer
```

Schrijf een instantie (instance) van de klasse `Eq` voor dit type waarbij breuken gelijk zijn volgens de normale wiskundige definities. Bijvoorbeeld twee derde ($\frac{2}{3}$) is gelijk aan vier zesde ($\frac{4}{6}$).

Opgave 9 *(20 punten)*

a) Gegeven zijn onderstaande definities van diverse functies. Bewijs met inductie dat de volgende gelijkheid geldt:

```
foldr f u .  map g = foldr (f .  g) u
```

Schrijf per herschrijfstap de rechtvaardiging op. *(16 punten)*

```
map :: (a -> b) -> [a] -> [b]
map f []     = []
map f (x:xs) = f x : map f xs

(.) :: (b -> c) -> (a -> b) -> (a -> c)
(.) f g x = f (g x)

foldr :: (a -> b -> b) -> b -> [a] -> b
foldr op unit []     = unit
foldr op unit (xLxs) = x `op` foldr op unit xs
```

b) Leg in hooguit 4 regels uit waarom een wet zoals bovenstaande handig kan zijn. *(4 punten)*