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)
18 februari 2004

Opgave 1: Types bepalen

a) Wat is het type van de functie `diff`, gedefinieerd door *(5 punten)*

```
diff a b = a - b
```

b) Schrijf het type op van de onderstaande expressie. Een afleiding is niet nodig. Als de expressie een typeringsfout bevat, schrijf dan op "typeringsfout". *(5 punten)*

```
map (++[0]) [[1],[2,1],[3,2,1]]
```

Opgave 2: Expressies evalueren

Hieronder staat telkens een expressie waarvan je de waarde moet bepalen. Als de expressie een typeringsfout bevat, antwoord dan met *D*. Er is steeds precies 1 antwoord goed.

a) `foldl (-) 20 [1, 2, 3]` *(5 punten)*

```
A.  -18
B.   14
C.  -24
D.  typeringsfout
```

b) `let f x = even ( x + 1 )` *(5 punten)*
`in map f [1, 2, 3, 4]`

```
even x = x `mod` 2 == 0
```

```
A.  [2, 4]
B.  [True, Fals, True, False]
C.  [1,  3]
D.  typeringsfout
```

c) `let p x = 3 > x in filter p [1, 2, 3, 4]` *(5 punten)*

```
A.  [1, 2]
B.  [True, True, False, False]
C.  [4]
D.  typeringsfout
```

Opgave 3: Haakjes zetten

Bij iedere vraag staat één expressie vet gedruk. Daaronder staan drie expressies die alleen maar verschillen in hoeveelheid en plaatsing van ronde haakjes. Kies die expressie die dezelfde betekenis heeft als de vetgedrukte expressie. Als geen expressie overeenkomt, kies dan *D*. Er is steeds precies 1 antwoord goed.

a) `(+) (f x y) (sin(z))` *(5 punten)*

```
A.  + (f x y) (sin z)
B.  (+) f x y (sin z)
C.  (+) (f x y) (sin z)
D.  geen van de bovenstaande drie
```

b) `map f xs ++ g ys` *(5 punten)*

```
A.  map f (xs ++ g ys)
B.  (map f xs) ++ (g ys)
C.  map f xs (++) g ys
D.  geen van de bovenstaande drie
```

Opgave 4: Functies schrijven

a) Schrijf op twee manieren een functie die bepaalt hoe vaak een letter in een string voorkomt. Een voorbeeld van gebruik: *(25 punten)*

```
Toets1> tel 'a' ['a', 'b', 'c', 'a']
2
```

Schrijf deze functie eerst zonder gebruik te maken van standaardfuncties, dus met expliciete recursie:

```
telRec :: Char -> String -> Int
```

En dan door juist wel gebruik te maken van standaardfuncties (zoals map en filter) en zonder recursie:

```
telStFun :: Char -> String -> Int
```

b) Schrijf de definitie *en het type* van de operator `<<` die bepaalt of de rechter parameter (een string) begint met de linker parameter: *(20 punten)*

```
Toets1> [1,2] << [1,2,3]
True
Toets1> [1,2], <<[1,3,3]
False
```

c) Schrijf map met behulp van foldr. Oftewel vul in de volgende definitie op de plekken waar drie puntjes staan de juiste Haskellcode in: *(20 punten)*

```
map :: (a -> b) -> [a] -> [b]

map f xs = let ... in foldr ...
```