Algoritmen en Programmeertalen
Tentamen Programmeertalen IN1605 II
28 augustus 2009 9.00-12.00

Afdeling ST **Faculteit EWI** **TU Delft**

Toegestaan is het gebruik van The craft of Functional Programming, een C- boek naar keuze en de college-slides.

Opgave 1 (15 punten)

Hieronder is de file stack.h gegeven. Samen met de file stack.c levert deze file een stack-object, waarin getallen kunnen worden bewaard.

```
#include<stdlib.h>
#include<stdio.h>

int empty();
int pop();
void push (int waarde);
```

De stack wordt geimplementeerd als een gelinkte lijst. De file stack.c, die deze implementatie bevat, begint aldus:

```
#include "stack.h"
struct node{
  int content;
  struct node * next;
};

struct node * stack = NULL;
```

Geef de volledige implementatie van de file stack.c.

Opgave 2 (20 punten)

Bepaal van elk van de volgende expressies het meest algemene type:

a. `[(< "type")]`

b. `map curry`

c.
```
apply where apply (f, x) = f x
```

d.
```
until
  where
    until p f x
      | p x       = x
      | otherwise = until p f (f x)
```

e. `Knoop "knoop"`

gegeven de definitie:

```
data Boom t = Blad t | Knoop t (Boom t) (Boom t)
```

Opgave 3 (10 punten)

Binnen een programma is behoefte aan een union type, dat soms een String moet bevatten, en soms een double. Implementeer zo'n datatype in

a. Java
b. Haskell

Opgave 4 (25 punten)

Geef de waarde van elk van de volgende expressies:

a.
```
let
    belasting inkomen
      | inkomen < 2000  =  inkomen / 10
      | inkomen < 4000  =  inkomen / 5
      | inkomen < 8000  =  inkomen / 4
      | otherwise       =  inkomen / 2
    in belasting 2000 + belasting 8000
```

b.
```
let
    m y [] = [y]
    m y (x:xs)
        | y == x    = xs
        | otherwise = x : m y xs
    in m 3 [1,3,2,3,1]
```

c.
```
((\x -> 10 - x). (*2). pred) 3
```

d.
```
foldr (+) 1 [2..4]
```

e.
```
let ds = filter. elem
    in ds 's' ["banaan", "ananas"]
```

Opgave 5 (20 punten)

Implementeer in Haskell de volgende functies:

a. De functie `partsums :: [Int] -> [Int]`
   partsums geeft de partiële sommen van een rij getallen:
   `partsums [2, 3, 5, 7] = [2, 5, 10, 17]`

b. De functie `flatten :: Hierarchy t -> [t]`
   Het type Hierarchy definieert een algemene hiërarchie:
   `data Hierarchy t = Leaf t | Node [Hierarchy t]`
   De functie `flatten` zet alle elementen uit de hiërarchie in een lijst.