Algoritmen en Programmeertalen
Tentamen Programmeertalen IN1605 II
28 juni 2010 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 (10 + 15 punten)
Een dubbel gelinkte lijst, circulaire lijst wordt in C geïmplementeerd met behulp van een listnode:

```
#include <stdio.h>

struct listnode{
     int value;
     struct listnode * next;
     struct listnode * previous;
};

typedef struct listnode * List;
```

Implementeer in C op basis van deze definities de functies:

a) `List add (List l, int a)`
Deze procedure voegt het element a in, na de node waar l naar wijst,
en geeft de aangepaste lijst terug, als verwijzing naar de node die a bevat.
Houd rekening met een lijst die aanvankelijk leeg is (`l == NULL`).

b) `List remove(List l)`
Deze procedure verwijdert de node waar l naar wijst uit de lijst,
en geeft de aangepaste lijst terug, met een verwijzing naar de voorganger van de verwijderde node. Als het verwijderde element het laatste is, moet een null-pointer worden teruggegeven.

Opgave 2 (20 punten)

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

a) `[head "head"]`
b) `id tail`
c) `[fst, snd]`
d)
```
zolang (< 'Z')
  where  zolang p f x
           | p x       = zolang p f (f x)
           | otherwise = x
```
e) Naam
gegeven de definities:
```
data Numexp = Naam Varnaam |
              Getal Int    |
              Samengesteld Numop Numexp Numexp
data Numop  = Plus | Min | Maal | Mod | Div
```

Opgave 3 (25 punten)

Geef de waarde van elk van de volgende expressies:

a) `(pred . (+4). (\ x -> 2 * x)) 5`

b)
```
let diffs xs = [y - x | x <- xs, y <- xs, x < y]
   in length (diffs [1,2,3,2])
```

c)
```
let f x = case x of
            (1:2:3:[4]) -> 1
            (1:2: )     -> 2
            (1:_)       -> 3
            (_)         -> 4
   in f [1..5]
```

d) `(unwords . map reverse . words) "aap noot Mies"`

e) `foldr (+) 2 [3..5]`

Opgave 4 (10 + 10 punten)

Implementeer in Haskell de volgende functies:

a) De functie `select :: [Int] -> [t] -> [t]` heeft als argument een lijst van indices, en een tweede lijst. De bedoeling van deze functie is de lijst op te leveren die ontstaat uit de index-lijst door elke index te vervangen door het bijbehorende element uit de tweede lijst.

```
select [6,4,2] [1..10] => [7,5,3]
select [5,4,2,1] "stroop" => "port"
```

b) De functie `size :: Hierarchy t -> Int`

Het type Hierarchy definieert een algemene hiërarchie:

```
data Hierarchy t = Leaf t | Node [Hierarchy t]
```

De functie size telt het aantal elementen uit de hiërarchie.