Algoritmen en Programmeertalen
Tentamen Programmeertalen IN1605 II
22 juni 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 (3 + 5 + 9 + 8 punten)
Een gelinkte lijst wordt in C geïmplementeerd met behulp van een listnode:

```
#include <stdio.h>

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

typedef struct listnode * List;
```

Voor de laatste node van de lijst geldt: `next == NULL`
Implementeer in C op basis van deze definities de functies:

a) `int head(List l)`
Deze functie geeft de waarde van het eerste element van de lijst terug.

b) `int last(List l)`
Deze functie geeft de waarde van het laatste element van de lijst terug.

c) `List remove(int a, List l)`
Deze procedure verwijdert alle voorkomens van a uit de lijst,
en geeft de aangepaste lijst terug.

d) `List nub (List l)`
Deze procedure verwijdert alle duplicaten uit de lijst,
en geeft de aangepaste lijst terug.

Opgave 2 (20 punten)

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

a) `["H"]`
b) `tail.id`
c) `flip head`
d) `[(head x, x)]`
e) `Samengesteld Plus`
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)
```
(succ . (2+). (\ x -> 3 * x)) 4
```

b)
```
let v = [(x, 2*x) | x <- [1..10]] in
    [2 * x | (_,x) <- v , x < 10]
```

c)
```
turtle "ZOON" (0,0)
  where
  turtle s (x,y) = case s of
                     ( []     ) -> (x,y)
                     ('N' :xs) ->  turtle xs (x, y + 1)
                     ('O' :xs) ->  turtle xs (x + 1, y)
                     ('Z' :xs) ->  turtle xs (x, y - 1)
                     ('W' :xs) ->  turtle xs (x - 1, y)
```

d)
```
(sum . map length) ["aap", "noot", "Mies"]
```

e)
```
foldr (==)"aap" ["noot", "mies", "wim"]
```

Opgave 4 (10 + 10 punten)

Implementeer in Haskell de volgende functies:

a) De functie `exists :: (a -> Bool) -> [a] -> Bool` heeft een predikaat en een lijst als argument, en gaat na of de lijst een element bevat dat aan het predikaat voldoet.
```
exists (<3) [1..5]  => True
exists (<3) [6..10] => False
```

b) De functie:
`namen:: Numexp -> [String]`
die een lijst teruggeeft van alle namen die in de Numexp voorkomen. Een naam mag maar eenmaal in de lijst voorkomen, maar u mag de library functie nub gebruiken.
Het type Numexp is als volgt gedefinieerd:
```
data Numexp = Naam String |
              Getal Int   |
              Samengesteld Numop Numexp Numexp
data Numop  = Plus | Min | Maal | Mod | Div
```