

## Lösung Übungsblatt Nr. 4 / ALP 1 zur Abgabe 11.12.2003

### Aufgabe 1:

#### Programmcode

```
split:: [Int] -> ([Int],[Int])
split [] = ([],[])
split [foobar] = ([foobar],[])
split (firstuneven:(firsteven:(restoflist))) = ((firstuneven:uneven),(firsteven:even))
  where (uneven,even) = split restoflist

combine:: ([Int],[Int]) -> [Int]
combine ([],foobar) = foobar
combine (foobar,[]) = foobar
combine ((x:xs),(y:ys))
  | x >= y = [x] ++ combine ((xs),(y:ys))
  | otherwise = [y] ++ combine ((x:xs),(ys))

mergesort:: [Int] -> [Int]
mergesort [] = []
mergesort [foobar] = [foobar]
mergesort reallist = combine (mergesort (uneven),mergesort (even))
  where (uneven,even) = split reallist
```

#### Testlauf

```
Main> mergesort [1..10]
[10,9,8,7,6,5,4,3,2,1]
Main> mergesort [0,7,43,28,654,0,24,234,6,456,5465]
[5465,654,456,234,43,28,24,7,6,0,0]
Main> mergesort [1]
[1]
```

#### Beschreibung

Die Funktion `split` bekommt eine Liste übergeben und gibt zwei Listen als Zweiertupel aus. (Listen vom Typ `Integer`)

Fallunterscheidung: Sie bekommt eine

- leere Liste - Ausgabe: zwei leere Listen als Zweiertupel.
- einelementige Liste - Eine leere und eine einelementige Liste als Zweiertupel.
- "echte" Liste übergeben: Sie nimmt die ersten beiden Elemente und legt sie jeweils als `firsteven:even` und `firstuneven:uneven` in das Zweiertupel, wobei `even` und `uneven` die Listen darstellen, welche durch den Aufruf `split restoflist` erzeugt werden.

Die Funktion `combine` erhält zwei Listen als Zweiertupel und gibt eine Liste aus. (Listen vom Typ `Integer`)

Falls sie eine leere Liste und eine einelementige Liste (oder umgekehrt) bekommt, dann gibt sie nur die einelementige Liste zurück. Ansonsten Sie vergleicht die beiden ersten Elemente der gegebenen Listen aus dem Tupel:

Falls das erste Element aus der ersten Liste größer dem ersten aus der zweiten Liste ist, liefert sie das erste Element der ersten Liste ++ `combine` (Rest der ersten Liste, Komplette zweite Liste) zurück.

Anderenfalls liefert sie das erste Element der zweiten Liste ++ `combine` (komplette erste Liste, Rest der zweiten Liste) zurück.

Die Funktion `mergesort` bekommt eine Liste vom Typ `Integer`, und gibt eine ebensolche zurück.

Falls Sie eine leere oder einelementige Menge als Eingabe bekommt, ist die Ausgabe gleich der Eingabe.

Falls Sie eine Liste mit mehr als einem Element bekommt, mischt Sie die Ausgaben der Funktion `mergesort (uneven)` und `mergesort (even)`, wobei `uneven` und `even` durch `split` (Eingabeliste) ermittelt werden.

## Lösung Übungsblatt Nr. 4 / ALP 1 zur Abgabe 11.12.2003

### Aufgabe 2:

#### Programmcode

g steht für gebunden, f steht für frei.

#### 2a)

$((\lambda z.zxx)zx) = ((\lambda g.gff)ff)$

#### 2b)

$((\lambda x.(\lambda x.(\lambda x.x)x)x)x) = ((\lambda g(\lambda g(\lambda g.g)g)g)f)$

#### 2c)

$(\lambda x.y(\lambda y.x(\lambda x.xz(\lambda y.yx)))) = (\lambda g.f(\lambda g.g(\lambda g.gf(\lambda g.gg))))$

#### 2d)

$(\lambda x.(xy(\lambda y.((xy)(\lambda x.wxyz))(\lambda z.wyz)))) = (\lambda g(\lambda gf(\lambda g.((gg)(\lambda g.fggf))(\lambda g.fgg))))$

### Aufgabe 3

a)

$((\lambda x.\lambda y.(y x)) \lambda p.\lambda q.p) \lambda i.i$   
->  $(\lambda y.(y \lambda p.\lambda q.p)) \lambda i.i$   
->  $(\lambda i.i \lambda p.\lambda q.p)$   
->  $(\lambda p.\lambda q.p)$

b)

$((((\lambda x.\lambda y.\lambda z.(x y) z)) \lambda f.\lambda a.(f a)) \lambda i.i) \lambda j.j$   
->  $((((\lambda x.\lambda y.\lambda u.(x y) z)) \lambda f.\lambda a.(f a)) \lambda i.i) \lambda j.j$   
->  $((((\lambda y.\lambda u.(z y)) \lambda f.\lambda a.(f a)) \lambda i.i) \lambda j.j)$   
->  $((\lambda u.(z \lambda f.\lambda a.(f a)) \lambda i.i) \lambda j.j)$   
->  $(z \lambda f.\lambda a.(f a)) \lambda j.j$

c)

$(\lambda h.((\lambda a.\lambda f.(f a)) h) h) \lambda f.(f f)$   
->  $(\lambda h.((\lambda f.(f h)) h) \lambda f.(f f))$   
->  $(\lambda h.(h h)) \lambda f.(f f)$   
->  $\lambda f.(f f) \lambda f.(f f)$

d)

$((((\lambda f.\lambda g.\lambda x.(f (g x))) \lambda s.(s s)) \lambda a.\lambda b.b) \lambda x.\lambda y.x)$   
->  $((((\lambda g.\lambda x.(\lambda s.(s s) (g x))) \lambda a.\lambda b.b) \lambda x.\lambda y.x)$   
->  $((\lambda g.\lambda x.((g x) (g x))) \lambda a.\lambda b.b) \lambda x.\lambda y.x)$   
->  $(\lambda x.((\lambda a.\lambda b.b x) (\lambda a.\lambda b.b x))) \lambda x.\lambda y.x)$   
->  $(\lambda x.((\lambda b.b) (\lambda b.b))) \lambda x.\lambda y.x)$   
->  $(\lambda b.b) (\lambda b.b)$   
->  $(\lambda b.b)$

e)

$((\lambda p.((\lambda q.(p q))((\lambda x.x) \lambda a.\lambda b.a))) \lambda k.k)$   
->  $((\lambda p.((\lambda q.(p q))(\lambda a.\lambda b.a))) \lambda k.k)$   
->  $((\lambda p.((p (\lambda a.\lambda b.a))) \lambda k.k)$   
->  $(\lambda k.k (\lambda a.\lambda b.a))$   
->  $(\lambda a.\lambda b.a)$

### Aufgabe 4

XOR/Antivalenz:

$XOR a b = NOT(a \leq b) = NOT(OR (AND a b) (AND NOTa NOTb))$

Es gilt:

$NOT = \lambda x.xFT$

$AND = \lambda uv.uvF$

$OR = \lambda wz.wTz$

$T = \lambda gh.g$

$F = \lambda km.m$

$Tab = a$

$Fab = b$

Identität =  $\lambda a.a$

Daraus folgt:

$(\lambda x.x) (\lambda uv.(\lambda uv.(\lambda u.uFT) (\lambda v.vFT) F) T (\lambda uv.uvF) FT)$