TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Lehrstuhl für Sprachen und Beschreibungsstrukturen SoS 2007
A. Berlea, F. Forster, T. Gawlitza Lösungsvorschläge zu Blatt 4
Ausgegeben am: 25. Mai 2007
Abgabe bis: **8. Juni 2007**

Übungen zu Einführung in die Informatik II

Aufgabe 1 Einfache Funktionen (Lösungsvorschlag)

a)
```
let betrag a =
  if a < 0 then (-a)
  else a
```

b)
```
let signum a =
  if a > 0 then 1
  else if (a < 0) then -1
  else 0
```

c)
```
let min a b =
  if a < b then a
  else b
```

d)
```
let four_times_h h x = h(h(h(h x)))
```

e)
```
let rec n_times_h h x n =
  if n = 0 then x
  else h (n_times_h h x (n-1))
```

alternativ:

```
let rec n_times_h h x n =
  if n = 0 then x
  else n_times_h h (h x) (n-1)
```

f)
```
let g_n_times_h h x g n = n_times_h h x (g n)
```

Aufgabe 2 Mengen als Listen (Lösungsvorschlag)

```
let find e l =
  match l with
    [] -> false
  | h::r -> if e=h then true
            else find e r

let rec union l1 l2 =
  match l1 with
    [] -> l2
  | h::r -> if find h l2 then union r l2
            else h::(union r l2)

let rec intersection l1 l2 =
  match l1 with
    [] -> []
  | h::r -> if find h l2 then h::(intersection r l2)
            else intersection r l2

let rec difference l1 l2 =
  match l1 with
    [] -> []
  | h::r -> if find h l2 then difference r l2
            else h::(difference r l2)
```

Aufgabe 3 Mengen als sortierteListen (Lösungsvorschlag)

```
let findSorted e l =
  let rec doit l = match l with
                     [] -> false
                   | h::r -> if e=h then true
                                 else if e<h then false
                      else doit r
  in doit l

let rec unionSorted l1 l2 =
  match (l1,l2) with
    ([],_) -> l2
  | (_,[]) -> l1
  | (h1::r1,h2::r2) -> if (h1<h2) then h1::( unionSorted r1 l2)
                       else if (h1>h2) then h2::( unionSorted l1 r2)
                else h1::(unionSorted r1 r2);;

let rec intersectionSorted l1 l2 =
  match (l1,l2) with
    ([],_) -> []
  | (_,[]) -> []
  | (h1::r1,h2::r2) -> if (h1=h2) then h1::(intersectionSorted r1 r2)
                       else if h1<h2 then intersectionSorted r1 l2
                    else intersectionSorted l1 r2
```

```
let rec differenceSorted l1 l2 =
  match (l1,l2) with
    ([],_) -> []
  | (_,[]) -> l1
  | (h1::r1,h2::r2) -> if (h1<h2) then h1::(differenceSorted r1 l2)
                       else if h1=h2 then differenceSorted r1 r2
                    else differenceSorted l1 r2
```

Aufgabe 4 Finanzberater (Lösungsvorschlag)

```
let rs (z,r,b,m) =
  let rec doit rs n =
    if rs < 0.0 then 0.0
    else if n<=0 then rs
    else doit (rs +. rs *. z -. r) (n-1)
  in doit b m

let rs2 z r b m =
  rs (z,r,b,m)

let berater1 = rs2 0.004
let berater2 = rs2 0.0035
let berater3 = berater1 1000.0
let berater4 = berater1 2000.0
```

Aufgabe 5 Mergesort (Lösungsvorschlag)

```
let init l = List.map (fun x -> [x]) l

let rec merge l1 l2 =
    match (l1,l2) with
        ([],y) -> y
      | (x,[]) -> x
      | (x::xs,y::ys) ->
            if x <= y then
                x::(merge xs (y::ys))
            else
                y::(merge (x::xs) ys)

let rec merge_list = function
    [] -> []
  | [l] -> [l]
  | l1::l2::ls -> (merge l1 l2)::(merge_list ls)

let mergesort l =
    let l = init l in
    let rec doit l =
        match l with
            [] -> []
          | [x] -> x
          | _ -> doit (merge_list l)
    in
    doit l
```