Programmierung WS 2002 / 03:
Musterlösung zum 9. Übungsblatt

Prof. Dr. Gert Smolka, Dipl.-Inform. Thorsten Brunklaus

Aufgabe 9.1: Endliche Mengen mit Gleicheit ($5 + 10$**)**

(a)
```
signature ISET' =
  sig
    type set
    val  empty  : set
    val  insert : int * set -> set
    val  member : int * set -> bool
    val eq      : set * set -> bool
  end
```

(b)
```
structure ISet' :> ISET' =
  struct
    type set = int list
    val empty = nil
    fun insert (i, nil)   = [i]
      | insert (i, x::xr) = if i<x then i::x::xr
                            else if i=x then x::xr
                            else x::(insert(i, xr))
    fun member (i, nil)   = false
      | member (i, x::xr) = if x<i then member(i, xr)
                            else i=x
    fun eq(xs,ys) = xs = ys
  end
```

Aufgabe 9.2: Endliche Funktionen ($5 + 10 + 5 + 5 + 10 + 10 + 5$**)**

(a)
```
signature IMAP =
  sig
    type 'a map
    val empty  : 'a map
    val insert : int * 'a * 'a map -> 'a map
    val lookup : int * 'a map -> 'a option
  end
```

(b)
```
structure IMap :> IMAP =
  struct
    type 'a map = int -> 'a option
    val empty  = (fn _ => NONE)
    fun insert (k,a,m) = (fn k' => if k=k' then SOME a else m k')
    fun lookup (k, m)  = m k
  end
```

(c)
```
val m = IMap.insert
        (1, 1, IMap.insert
         (5, 3, IMap.insert
          (6, 0, IMap.empty)))
```

(d)
```
type 'a map = 'a IMap.map
val empty  = IMap.empty
val insert = IMap.insert
val lookup = IMap.lookup
```

(e)
```
signature ISET =
  sig
    type set
    val  empty  : set
    val  insert : int * set -> set
    val  member : int * set -> bool
  end

structure ISet :> ISET =
  struct
    type set = unit IMap.map
    val empty = IMap.empty
    fun insert (n,s) = IMap.insert(n,(),s)
    fun member (n,s) = isSome(IMap.lookup(n,s))
  end
```

(f)
```
functor Map
  (type key
   val  compare : key * key -> order)
  :>
  sig
    type 'a map
    val empty  : 'a map
    val insert : key * 'a * 'a map -> 'a map
    val lookup : key * 'a map -> 'a option
  end
  =
  struct
    type 'a map = key -> 'a option
    val empty  = (fn _ => NONE)
    fun insert (k,a,m) =
      (fn k' => if compare(k,k')=EQUAL then SOME a else m k')
    fun lookup (k, m)  = m k
  end
```

(g)
```
structure IMap = Map
  (type key = int
   val compare = Int.compare)
```

Aufgabe 9.3: Priorisierte Schlangen ($5 + 10 + 5 + 10 + 5$**)**

(a)
```
signature IPQUEUE =
  sig
    type 'a pqueue
    val empty  : 'a pqueue
    val insert : int * 'a * 'a pqueue -> 'a pqueue
    val head   : 'a pqueue -> int * 'a   (* Empty *)
    val tail   : 'a pqueue -> 'a pqueue  (* Empty *)
  end
```

(b)
```
structure IPQueue :> IPQUEUE =
  struct
    type 'a pqueue = (int * 'a) list
    val empty = nil
    fun insert (k, a, nil)       = [(k,a)]
      | insert (k, a, (l,b)::es) = if k<l
                                   then (k,a)::(l,b)::es
                                   else (l,b)::insert(k,a,es)
    val head = hd
    val tail = tl
  end
```

(c)
```
open IPQueue

val q = insert
         (2, "Tom", insert
          (3, "Monica", insert
           (2, "Maria", insert
            (4, "Jim", empty))))
```

(d)
```
functor PQueue
  (type key
   val  compare : key * key -> order)
  :>
  sig
    type 'a pqueue
    val empty  : 'a pqueue
    val insert : key * 'a * 'a pqueue -> 'a pqueue
    val head   : 'a pqueue -> key * 'a  (* Empty *)
    val tail   : 'a pqueue -> 'a pqueue  (* Empty *)
  end
  =
  struct
    type 'a pqueue = (key * 'a) list
    val empty = nil
    fun insert (k, a, nil)       = [(k,a)]
      | insert (k, a, (l,b)::es) = if compare(k,l) = LESS
                                   then (k,a)::(l,b)::es
                                   else (l,b)::insert(k,a,es)
    val head = hd
    val tail = tl
  end
```

(e)
```
structure IPQueue = PQueue
  (type key = int
   val  compare = Int.compare)
```