



Programmierung: Musterlösung zum 9. Übungsblatt

Prof. Gert Smolka und Thorsten Brunklaus

Aufgabe 9.1: Endliche Funktionen (5+5+5+5+5+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

structure IMap :> IMap =
  struct
    type 'a map = (int * 'a) list
    val empty = nil
    fun insert (k,a,s) = (k,a)::s
    fun lookup (k, nil) = NONE
      | lookup (k, (l,a)::s) = if k=l then SOME a else lookup(k,s)
  end
```
- (b)

```
val m = IMap.insert
  (1, 1, IMap.insert
    (5, 3, IMap.insert
      (6, 0, IMap.insert
        (7, 0, IMap.empty))))
```
- (c)

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

```
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
```

```

(e) 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) list
      val empty = nil
      fun insert (n,x,s) = (n,x)::s
      fun lookup (n, nil) = NONE
        | lookup (n, (m,x)::s) = if compare(n,m)=EQUAL then SOME x
                                   else lookup(n,s)
    end

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

```

Aufgabe 9.2: Priorisierte Schlangen (5+5+5+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

    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

(b) open IPQueue

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

```

```

(c) 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

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

```