


```

        else (b,a,p)::(e xs)
    in Tabelle(e xs) end
fun loeschen (string,Tabelle xs) =
    let fun l [] = raise nicht_gefunden
        | l ((b,a,p)::xs) = if string = b
            then xs
            else (b,a,p)::(l xs)
    in Tabelle(l xs) end
fun find (string,Tabelle xs) =
    let fun f [] = raise nicht_gefunden
        | f ((b,a,p)::xs) = if string = b
            then (b,a,p)
            else f xs
    in f xs end
fun bezeichnung (string,tabelle) = #2(find(string,tabelle))
fun preis (string,tabelle) = #3(find(string,tabelle))
end;;

open TABELLE;;

val t1 = neue_tabelle[("123","Zahnpasta",2.39),
                      ("0815","Leberwurst",1.99),
                      ("836","Cornflakes",3.49),
                      ("9234","Kleinwagen",19999.99)];

(* Teilaufgabe (c) *)

signature VERKAUFSig =
sig
type verkauf
val neuer_verkauf : verkauf
val neuer_artikel : int * string * verkauf -> verkauf
val artikelliste : verkauf * tabelle -> (string * real * int * real) list
val gesamtpreis : verkauf * tabelle -> real
end;;

(* Teilaufgabe (d) *)

structure VERKAUF : VERKAUFSig =
struct
datatype verkauf = Verkauf of (int * string) list
val neuer_verkauf = Verkauf[]
fun neuer_artikel(anzahl,barcode,Verkauf xs) =
    let fun na [] = [(anzahl,barcode)]
        | na ((n,b)::xs) = if b = barcode
            then (n+anzahl,b)::xs
            else (n,b)::(na xs)
    in Verkauf(na xs) end
fun artikelliste (Verkauf xs, tabelle) =
    map (fn (n,b) => (bezeichnung(b,tabelle),preis(b,tabelle),

```

```

        n, real(n)*preis(b,tabelle)))
    xs
fun gesamtpreis (Verkauf xs, tabelle) =
    foldl (op +) 0.0 (map (fn (n,b) => real(n)*preis(b,tabelle)) xs)
end;;

open VERKAUF;;

val v1 = neuer_artikel(2,"123",neuer_artikel(3,"0815",neuer_verkauf));;
val v2 = neuer_artikel(4,"836",neuer_artikel(3,"123",v1));;

```

Aufgabe 11-3

Endliche Mengen

()

```

use "set.sml";

open SET;

(* Teilaufgabe a - Zunächst definieren wir die Funktion subset *)

fun subset(a, b) = if isemptyset(a) then true
    else
        let
            val c = any(a)
        in
            member(c, b) andalso subset(delete(c, a), b)
        end;;

(* Teilaufgabe b - Mit Hilfe der Funktion subset läßt sich die
    Funktion seteq wie folgt definieren: *)

fun seteq(a, b) = subset(a, b) andalso subset(b, a);;

(* Es gibt aber auch eine wesentlich effizientere Lösung: *)

fun seteq(a,b) = if isemptyset(a) then isemptyset(b)
    else
        let
            val c=any(a)
        in
            member(c, b) andalso seteq(delete(c,a),delete(c,b))
        end;;

(* Teilaufgabe c *)

fun intersect(a, b) = if isemptyset(a) orelse isemptyset(b) then emptyset
    else
        let
            val c=any(a)

```

```

        in
        if member(c, b) then
            insert(c, intersect(delete(c,a),b))
        else intersect(delete(c,a),b)
    end;;

```

(* Teilaufgabe d - für die Lösung benötigen wir die in der Vorlesung definierte Funktion, die die Vereinigung zweier Mengen berechnet *)

```

fun union(a,b) = if isemptyset a then b
                 else if isemptyset b then a
                 else
                     let
                         val c=any a
                     in
                         if member(c, b) then union(delete(c,a),b)
                         else insert(c, union(delete(c,a),b))
                     end;;

```

```

fun cartprod(a,b) = if isemptyset a orelse isemptyset b then emptyset
                   else
                       let
                           val c=any a
                           val d=any b
                       in
                           insert((c,d),
                                union(cartprod(delete(c,a),b),
                                       cartprod(a, delete(d,b))))
                       end;;

```

Aufgabe 11-4

Schlangen

()

(* Teilaufgabe a *)

```

signature QUEUESig =
sig
    type 'a queue
    val emptyqueue   : 'a queue
    val isemptyqueue : 'a queue -> bool
    val first        : 'a queue -> 'a
    val dequeue      : 'a queue -> 'a queue
    val enter        : 'a * 'a queue -> 'a queue
end;;

```

(* Teilaufgabe b *)

```

structure QUEUE : QUEUESig =
struct

```

```

type 'a queue = 'a list
val emptyqueue = nil
fun isemptyqueue l = null(l)
fun enter(a, q) = if isemptyqueue(q) then [a]
                  else hd(q)::enter(a, tl(q))
fun dequeue l = tl l
fun first l = hd l
end;;

open QUEUE;;

val q1 = enter(4,enter(3,enter(2,enter(1,emptyqueue))));;
val q2 = enter(~3,enter(7,enter(~5,q1))));;

(* Teilaufgabe c *)

fun posqueue q =
  let fun map q acc = if isemptyqueue q then acc
                    else let val h = first q
                        in map (dequeue q)
                          (if h >= 0
                           then enter(h, acc)
                           else acc) end
  in map q emptyqueue end;;

(* Teilaufgabe d *)

fun revqueue q = if isemptyqueue(q) then q
                else enter(first(q), revqueue(dequeue(q))));;

(* Teilaufgabe e *)

fun foldlqueue f b q = if isemptyqueue(q) then b
                      else foldlqueue(f)(f(first(q), b))(dequeue(q));;

```