Klausur Informatik 2 , 15.02.2010

Lösung:

Aufgabe 1:

a)
```
public synchronized void acquireAirstripAndGate (int airplaneId) {
      while (!gate1.occupy(airplaneId)) {
            if (gate2.occupy(airplaneId)) {
                  break;
            }
            try { wait(); } catch (InterruptedException e) {}
      }
      while (!airstrip.occupy(airplaneId)) {
            try { wait(); } catch (InterruptedException e) {}
      }
}
public synchronized void acquireAirstrip (int airplaneId) {
      while (!airstrip.occupy(airplaneId)) {
            try { wait(); } catch (InterruptedException e) {}
      }
}
public synchronized void releaseAirstrip (int airplaneId) {
      airstrip.release(airplaneId);
      notifyAll();
}
public synchronized void releaseGate (int airplaneId) {
      if(!gate1.release(airplaneId)) {
            gate2.release(airplaneId);
      }
      notifyAll();
}
```

b)

```
private void touchdown() throws InterruptedException {
      airport.acquireAirstripAndGate(airplaneId);
      sleep(500);
      airport.releaseAirstrip(airplaneId);
}
private void refueling() throws InterruptedException {
      sleep(2000);
      airport.acquireAirstrip(airplaneId);
}
private void takeOff() throws InterruptedException {
      sleep(800);
      airport.releaseAirstrip(airplaneId);
}
```

Aufgabe 2:

a) f [1;3;5;7] = [5;7]

b) ‘a list -> ‘a list list

c) ‘a -> (‘a -> int) -> int[/code]

Aufgabe 3

```
let everySecond l =
        let rec second a list = match list
                with [] -> []
                | x::xs -> if (a == true) then (x::second false xs) else (second true xs)
        in second false l
```

alternative:

```
let rec everySecond l =
        match l with
                | x::y::r -> y::(everySecond r)
                | _ -> []
```

```
b) let rec repeat f a = if (a >= 100) then 0
        else if ((f a) > 100) then 1
        else (1 + (repeat f (f a)))
```

```
c) exception NoSuchElement;
let rec last l = match l
        with [] -> raise NoSuchElement;
        | x::[] -> x
        | x::xs -> last xs
```

Aufgabe 4

a) n = 0

map1 f1 l1 = map1 f1 [] = match [] with [] -> [] | … =
  = []

map2 f1 l1 = map2 f1 [] = fold_right (first f1) [] [] = match [] with [] -> [] | … =
  = []

n -> n + 1 l1 = x::xs

map1 f1 l1 = map1 f1 x::xs = match x::xs with … | h::t -> (f1 h) :: (map1 f1 t) =
  = (f1 x) :: (map1 f1 xs)
  = (f1 x) :: (map2 f1 xs) nach I.V.

map2 f1 l1 = map2 f1 (x::xs) = fold_right (first f1) (x::xs) [] =
  = match x::xs with … | h::t -> (first f1) h (fold_right (first f1) t []) =
  = (first f1) x (fold_right (first f1) xs [])

  Bemerkung: (fold_right (first f1) xs []) = map2 f1 xs

  = first f1 x (map2 f1 xs) = (f1 x) :: (map2 f1 xs) q.e.d.

Aufgabe 5

```
module PolImp : Pol = struct
        type polynom = (int * float) list
        let coeff p n = match p
                with [] -> 0.0
                | (a, b)::xs –> if (a == n) then b else (coeff xs n)

        let rec pow b e = match e
                with 0 -> 1.0
                | x -> b *. (pow b (x – 1))

        let rec eval p x = match p
                with [] -> 0.0
                | (a,b)::xs -> b *. (pow x a) +. eval xs x

        let rec diff p = match p
                with [] -> []
                | (a,b)::xs -> if (a == 0) then (diff xs)
                                else ((a – 1), (float_of_int a) *. b) :: (diff xs)

end
```

Aufgabe 6

a) let rec send_list l = match l
with [] -> sync (send chan_u ())
| x::xs -> sync (send chan_i x);
send_list xs

b) let rec receive_list _ = select [
wrap (receive chan_u) (fun () -> []);
wrap (receive chan_i) (fun x -> x::receive_list ())
]

c) let server _ = match (List.length receive_list)
with 0 -> print_string “0”
| x -> print_ string (string_of_int x);
server ()

d) let _ =
let t = create server () in
send_list [1;2;3;4];
send_list [];
join t;
print_string “Done!”