2. Vertiefung der rekursiven Programmiertechnik

30.10.02

2.1 Einbettung linear rekursiver Funktionen über IN

Problem: Goldbachsche Vermutung
(Existenz ∞-vieler Primzahlzwilling?)

Gesucht: Funktion zur Berechnung ~~der ersten~~ Primzahlzwillinge von 1 bis n

Lösungsidee (Divide & Conquer)

- Formuliere eine Funktion zum Testen der Primzahleigenschaft
  Formuliere eine Funktion, die von 2 bis n jeweils prüft, ob (m, m+2) Primzahlzwilling ist.

(A)
```
let isprim n = (n >=2) & not( isdiv(n/2, n) )
```

Hilfsfunktion

```
let rec isdiv arg = match arg with
      | (k, n) when k <=1 -> false
      | (k, n) -> (n mod k = 0) || isdiv(k-1, n)
```

(B)
```
let rec zwillinge arg = match arg with
      | (m, n, ls) when m=n -> ls
      | (m, n, ls) when (isprim(m) & isprim(m+2)) -> zwillinge(m+1, n, (m, m+2)::ls
      | m, n, ls) -> zwillinge(m+2, n, ls)
```

```
let goldbach n = zwillinge (2, n, []);;
```


2.2 „Partielles" Sortieren von Listen

Problem: Gegeben sei eine Liste von Elementen, mit einer Ordnung (z. B. „<", ...)

Gesucht: Rekursive Funktion, die diejenigen Elemente einer Liste zurückgibt, die kleiner als ein bestimmtes Element e' sind.

Lösungsidee:

- Ist die Liste leer → gib leere Liste zurück
- sonst, vergleiche man das erste Element x der Liste mit e'
  - wenn x < e' → füge x an die Ergebnisliste an
  - sonst nicht

→ rufe die Funktion für den Rest der Liste auf

mit Einbettung:

```
let less (e, ls) = h ([], e, ls);;

let rec h arg = match arg with
  | (lz, e, ls) when ls = [] -> lz
  | (lz, e, x::ll) when x<e
                -> h (lz@[x], e, ll)
  | (lz, e, x:ll) -> h (lz, e, ll)
```