Technische Universität München
Fakultät für Informatik
R. Kuntschke, I. Nagy,
Dr. M. Schneider

SS 2006
Lösungsvorschläge zu Blatt 1
12. Mai 2006

Praktikum zu Grundlagen der Programmierung

Aufgabe 1 Funktionen in OCaml (Lösungsvorschlag)

a) **Definition von Funktionen:**

(i)
```
# let square x = x *. x;;
val square : float -> float = <fun>
# square 5.;;
- : float = 25.
```

(ii)
```
# let square_root x = sqrt x;;
val square_root : float -> float = <fun>
# square_root 25.;;
- : float = 5.
```

Alternativlösung:

```
# let square_root = sqrt;;
val square_root : float -> float = <fun>
# square_root 25.;;
- : float = 5.
```

(iii)
```
# let square_id x = square_root (square x);;
val square_id : float -> float = <fun>
# square_id 5.;;
- : float = 5.
```

(iv)
```
# let validate_id x = x = square_id(x);;
val validate_id : float -> bool = <fun>
# validate_id 5.;;
- : bool = true
```

b) **Zusammengesetzte Funktionen:**

(i)
```
# let diag_r a b = square_root(square a +. square b);;
val diag_r : float -> float -> float = <fun>
# diag_r 3. 4.;;
- : float = 5.
# diag_r 4. 5.;;
- : float = 6.4031242374328485
```

(ii)
```
# let diag_q a b c = square_root(square (diag_r a b) +. square c);;
val diag_q : float -> float -> float -> float = <fun>
# diag_q 3. 4. 5.;;
- : float = 7.0710678118654755
```

(iii)
```
# let diag_c a = diag_q a a a;;
val diag_c : float -> float = <fun>
# diag_c 5.;;
- : float = 8.6602540378443873
# diag_q 5. 5. 5.;;
- : float = 8.6602540378443873
```

c) **Partielle Funktionsanwendung:**

```
# let diag_q1 = diag_q 1.;;
val diag_q1 : float -> float -> float = <fun>
# diag_q1 3. 4.;;
- : float = 5.0990195135927845
# diag_q 1. 3. 4.;;
- : float = 5.0990195135927845
```

Alternativlösung:

```
# let diag_q1 b c = diag_q 1. b c;;
val diag_q1 : float -> float -> float = <fun>
# diag_q1 3. 4.;;
- : float = 5.0990195135927845
```

Aufgabe 2 Osterberechnung (Lösungsvorschlag)

Eine mögliche Implementierung könnte folgendermaßen aussehen:

```
type monat = Januar | Februar | März | April | Mai
             | Juni | Juli | August | September
             | Oktober | November | Dezember;;

type datum = Datum of (int * monat * int);;

let tabelle_m j = match j with
| j when (j > 1582 && j < 1700) -> 22
| j when (j > 1699 && j < 1900) -> 23
| j when (j > 1899 && j < 2200) -> 24
| j when (j > 2199 && j < 2300) -> 25
| _ -> failwith "Falsche Eingabe";;

let tabelle_n j = match j with
| j when (j > 1582 && j < 1700) -> 2
| j when (j > 1699 && j < 1800) -> 3
| j when (j > 1799 && j < 1900) -> 4
| j when (j > 1899 && j < 2100) -> 5
| j when (j > 2099 && j < 2200) -> 6
| j when (j > 2199 && j < 2300) -> 0
| _ -> failwith "Falsche Eingabe";;

let d j = (19 *(j mod 19) + tabelle_m j) mod 30;;

let e j = (2 * (j mod 4) + 4 * ( j mod 7) + 6 * (d j) + tabelle_n j) mod 7;;
```

```
let ostersonntag j = let d_plus_e = d j + e j
in if ((22 + d_plus_e)<32) then Datum(22 + d_plus_e, März, j)
else Datum (d_plus_e - 9, April, j);;
```

Aufgabe 3 Währungsrechner (Lösungsvorschlag)

Eine mögliche Implementierung könnte folgendermaßen aussehen:

```
let umrechnung euro code =
    if code = 0 then (euro *. 1.95583)
    else if code = 1 then (euro *. 1.27367)
    else if code = 2 then (euro *. 0.68439)
    else if code = 3 then (euro *. 142.150)
    else if code = 4 then (euro *. 1.56039)
    else euro
;;
```

Aufgabe 4 Primzahltest (Lösungsvorschlag)

a) Test, ob x eine ungerade Zahl ist:

```
# let odd x =
if ((x mod 2) == 1) then 1 else 0;;
```

b) Test, ob x von der Form $6n+1$ ist:

```
# let testA x =
if (((x - 1) mod 6) == 0) then 1 else 0;;
```

c) Test, ob x von der Form $6n-1$ ist:

```
# let testB x =
if (((x + 1) mod 6) == 0) then 1 else 0;;
```

d) Test, ob x eine Primzahl ist, d.h. ob x nur durch 1 und sich selbst teilbar ist:

```
# let rec primeTest x y =
if (y == 1) then 1
else if (y > 1) && (x mod y == 0) then 0
else primeTest x (y-1);;
```

e) Kombination der erstellten Funktionen zu einem schnellen Primzahltest:

```
# let rec prime x =
if (x <= 3) then 1
else if ((odd x) == 1) && (((testA x) == 1) || ((testB x) == 1)) then
primeTest x (int_of_float (floor (sqrt (float x))))
else 0;;
```