Fachrichtung 6.2 — Informatik
Universität des Saarlandes
Tutorenteam der Vorlesung Programmierung 1

Programmierung 1 (Wintersemester 2012/13)

Lösungsblatt 5

(Kapitel 5)

Aufgabe 5.1 (*Aufgabe aus dem Nachklausurtutorium zum Wintersemester 11/12*)
Schreiben Sie eine Prozedur $anzBuchst : str \to (char * int)\ list$, die aus einem String eine Liste erstellt, die zu jedem Kleinbuchstaben (Großbuchstaben dürfen der Einfachkeit halber wegfallen) angibt, wie oft er in dem Text vorkommt. Dabei sollen Großbuchstaben so wie sämtliche Sonder- und Leerzeichen ignoriert werden.
Tipp: Schreiben Sie keine Prozedur, die jeden Buchstaben einzeln ausliest! Versuchen Sie stattdessen mit den String-Befehlen, eine einfachere Lösung zu finden. Gehen Sie dabei wie folgt vor:

(a) Schreiben Sie eine *val*-Deklaration, die Ihnen eine $chr * int\ list$ erzeugt, die Ihnen ein Paar der Buchstaben von a bis z und 0en erzeugt.

(b) Schreiben Sie eine Prozedur $erhoehen : int- > chr * int\ list \to char * int\ list$, die für die Buchstaben a bis z (wobei $a = 0$, $b = 1$, ..., $z = 25$) die Anzahl des entsprechenden Buchstaben in Ihrer $char * int\ list$ um eins erhöht.

(c) Konstruieren Sie mit diesen zwei Hilfsprozeduren eine Prozedur $anzBuchst : string \to (char * int)\ list$, die für einen gegebenen String die oben beschriebenen Bedingungen erfüllt!

Achten Sie dabei ganz besonders auf die Typen Ihrer Prozeduren und Hilfsprozeduren und schreiben Sie sich zur Hilfe mit Bleistift dazu, welche Typen die genutzten Bezeichner und Ausdrücke haben.

Lösung 5.1:

```
fun anzBuchst xs = let val clist = explode (xs)
val dlist = map ord clist
val a = ord (#a)
val elist = map (fn x => x-a) dlist
val buchstlist = List.tabulate (26, fn x => (chr (a+x), 0))
fun erhoehen 0 ((a,n)::xr) = (a, n+1)::xr
| erhoehen m ((a,n)::xr) = (a,n) :: (erhoehen (m-1) xr)
in
foldl (fn (x,ys) => if x>=0 andalso x<=25 then (erhoehen x ys) else ys)
buchstlist elist
end
```

Aufgabe 5.2 (*Offizielle Aufgabe aus dem Wintersemester 11/12*)
VIVA ist noch nicht ganz zufrieden mit Ihrer Arbeit. Sie sollen den *Love-Generator* so erweitern, dass für einen Harem nicht mehr nur ein einzelner Wert, sondern eine nach Kompatibilität geordnete Liste von Paarungen zurückgegeben wird. Schreiben Sie also eine Prozedur

$loveGenSort : string\ list \to (string * string)\ list$

die für eine Liste von Namen zunächst alle möglichen Paarungen berechnet und diese anschließ gend anhand ihres LoveGen-Kompatibilität sortiert.

Lösung 5.2:

```
fun sort xs = <siehe Aufgabe 6.3>
fun compare ((s1, s2),(t1, t2)) = Int.compare(loveGen(s1, s2), loveGen(t1,t2))
fun cartesian nil = nil
| cartesian (x::xr) = map (fn y => (x,y)) xr @ cartesian xr
fun loveGenSort xs = sort compare (cartesian xs)
```

Aufgabe 5.3 (*Schriftliche Aufgabe aus dem Wintersemester 11/12*)
Überlegen Sie sich potentielle Minitest-Aufgaben zum Stoff dieser Woche. Die Aufgaben sollen dabei in kurzer Zeit verständlich und bearbeitbar sein (ca. 3 bis 5 Minuten pro Aufgabe) und gleichzeitig versuchen, das Verständnis des Stoffs weitreichend abzufragen. Erklären Sie, weshalb Ihre Aufgaben diese Kriterien erfüllen. Lösen Sie dann Ihre eigenen Aufgaben!

Aufgabe 5.4 (*Schriftliche Aufgabe aus dem Wintersemester 11/12*)
Stellen Sie sich vor, Sie gehen in die Mensa. Überlegen Sie sich, an welchen Stellen die bisher vorgestellten Konzepte und Algorithmen den Ablauf reibungsloser gestalten würden. Wo könnte man andere Informatikkonzepte einbringen, die Sie unabhängig von Programmierung 1 kennengelernt haben?