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

Programmierung 1 (Wintersemester 2012/13)

Lösungsblatt 14

(Kapitel 15 und 16)

Hinweis: Dieses Übungsblatt enthält von den Tutoren für die Übungsgruppe erstellte Aufgaben.

Die Aufgaben und die damit abgedeckten Themenbereiche sind für die Klausur weder relevant, noch irrelevant.

Aufgabe 14.1 (*Lineare Darstellung von Listen*)
Sie haben in Kapitel 15.10 eine Möglichkeit kennen gelernt, um Listen von Zahlen in einer Halde darstellen zu können. Auf diese können Sie die schon bekannten Prozeduren auf Listen aus Kapitel 4 anwenden.
Hinweis: Allozieren Sie außer für *tabulate* keinen neuen Speicherplatz, wenn dies nicht unbedingt notwendig ist !

- Schreiben Sie eine Prozedur $length : address \to int$, die zu einer angegebenen Adresse die Länge der angegebenen verketteten Liste bestimmt!

  ```
  fun length a =
  ```

- Schreiben Sie eine Prozedur $map : address \to (int -> int) \to int$, die auf die Liste einer angegebenen Adresse die Prozedur *map* anwendet!

  ```
  fun map a f =
  ```

- Schreiben Sie eine Prozedur $exists : address \to (int \to bool) \to bool$, die auf die Liste einer angegebenen Adresse die Prozedur $filter$ anwendet!

  ```
  fun exists f a =
  ```

- Schreiben Sie eine Prozedur $all : address \to (int \to bool) \to bool$, die auf die Liste einer angegebenen Adresse die Prozedur *all* anwendet!

  ```
  fun all f a =
  ```

- Schreiben Sie eine Prozedur $filter : address \to (int \to bool) \to int$, die auf die Liste einer angegebenen Adresse die Prozedur $filter$ anwendet!

  Tipp: Sie benötigen eine Hilfsprozedur, um sich das letzte gültige Argument merken zu können!

  ```
  fun filter f a =
  ```

- Schreiben Sie eine Prozedur $tabulate : (int * int \to int) \to address$, die für eine Zahl n und eine Prozedur *tabulate* anwendet!

  ```
  fun tabulate (n,f) =
  ```

- Schreiben Sie eine Prozedur $rev : address \to address$, die die Liste einer Adresse reversiert!

  ```
  fun rev a =
  ```

Lösung 14.1:

```
fun length a = if a = ~1 then 0 else 1+ length (sub a 1)

fun map a f = if a = ~1 then ~1 else (update a 0 (f (sub a 0)); map (sub a 1) f; a)

fun exists f a = if a = ~1 then false else f (sub a 0) orelse exists f (sub a 1)

fun tabulate (n, f) = putList (List.tabulate (n,f))

fun filter' f a last = if a= ~1  then ~1 else if f (sub a 0) then filter f (sub a 1) a
else  if sub a 1 = ~1 then (update last 1 ~1; last)
   else (update a 0 (sub (sub a 1) 0); update a 1 (sub (sub a 1) 1);  filter f a last)

fun filter f a = filter' f a a

fun all f a = if a = ~1 then true else f (sub a 0) andalso all f (sub a 1)

fun rev' a last = if a = ~1 then (last) else
(let val res = rev' (sub a 1) a in  (update a 1 last; res) end)

fun rev a = rev' a ~1
```