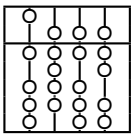
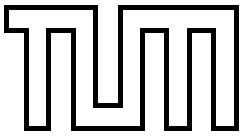


TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK



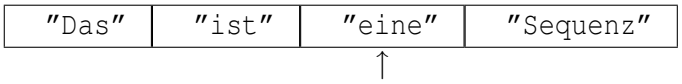
Lehrstuhl für Sprachen und Beschreibungsstrukturen
Prof. Dr. Helmut Seidl
A. Berlea, F. Forster, T. Gawlitza

SoS 2007
Übungsblatt 6
Ausgegeben am: 22. Juni 2007
Abgabe bis: **06. Juli 2007**

Übungen zu Einführung in die Informatik II

Aufgabe 1 (H) Datenstruktur (15 Punkte)

Ziel dieser Aufgabe ist es den nachfolgend skizzierten Datentyp `'a t` in OCaml zu implementieren. Ein Wert des Typs `'a t` soll eine Sequenz von Werten des Typs `'a` darstellen, wobei ein „Zeiger“ auf die **aktuelle Zelle** zeigt. Nachfolgende Abbildung illustriert die Situation.



Ihre Aufgabe ist es OCaml-Funktionen zu definieren, so dass man den Wert der aktuellen Zelle auslesen, Elemente hinzufügen, sowie den Zeiger nach vorne und hinten bewegen kann. Definieren Sie

a) einen geeigneten Typ `'a t`.
b) eine geeignete `to_string`-Funktion.
c) einen Wert `empty : 'a t`, der eine leere Sequenz darstellt.
d) eine Funktion `is_empty : 'a t -> bool`, die `true` zurück liefern genau dann, wenn die übergebene Sequenz leer ist.
e) Funktionen `is_first: 'a t -> bool` und `is_last: 'a t -> bool`, die true zurück liefern genau dann, wenn der Zeiger auf dem ersten bzw. auf dem letzten Element steht.
f) eine Funktion `len : 'a t -> int`, die die Anzahl der Elemente zurück liefert.
g) Funktionen `next : 'a t -> 'a t` und `prev: 'a t -> 'a t`, die eine neue Sequenz zurück liefert, bei der der Zeiger um eine Zelle nach rechts bzw. nach links versetzt ist.
h) eine Funktion `add : 'a t -> 'a -> 'a t`. Ein Aufruf `add s x` soll aus der Sequenz `s` die Sequenz erzeugen, bei der das Element `x` an der Zeiger-Position hinzugefügt wurde.
i) eine Funktion `add_before : 'a t -> 'a -> 'a t`. Ein Aufruf `add_before s x` soll aus der Sequenz `s` die Sequenz erzeugen, bei der das Element `x` eine Zelle vor der Zeiger-Position hinzugefügt wurde.
j) eine Funktion `move : 'a t -> int -> 'a t`. Ein Aufruf `move s pos` bewegt den Zeiger an die Position `pos`.

Achten Sie bei Ihrer Implementierung auf Effizienz.

Hinweise:

a) Schauen Sie sich die Implementierung der Datenstruktur **Queue** aus der Vorlesung an.
b) Sie können die in den Modulen `List` und `String` bereitgestellte Funktionalität nutzen.

Aufgabe 2 (H) Powerpoint **(15 Punkte)**

Ziel dieser Aufgabe ist es Datentypen und Funktionen zur Repräsentation von Präsentationen zu erstellen. Eine Präsentation besteht aus mehreren Folien. Jede Folie hat einen Titel (Zeichenkette), eine optionale Kopf- und eine optionale Fußzeile sowie einen Inhalt. Letztere können aus mehreren Absätzen bestehen. Ein Absatz baut sich wiederum aus mehreren Elementen auf, wobei folgende Elemente möglich sind:

- Tabelle: eine zweidimensionale Matrix von Absätzen
- Itemize-Umgebung: eine Folge von Absätzen
- Bild: eine Zeichenkette, die den Pfad zum Bild angibt
- Link: eine Zeichenkette, die eine URL angibt
- Wort: eine Zeichenkette

a) Definieren Sie geeignete Datentypen zur Repräsentation von Präsentationen.

   (Hinweis: Zur Darstellung optionaler Werte kann der vordefinierte Datentyp `'a option` benutzt werden.)

b) Schreiben Sie eine Funktion, die die Folien einer Präsentation durchnummeriert.

c) Schreiben Sie eine Funktion, die einen Index aller vorkommenden Wörter erstellt und ausgibt. Der Index soll zu jedem Wort die Liste der Nummern der Folien, in denen das Wort vorkommt, enthalten.

d) Schreiben Sie eine Funktion, die den gesamten Inhalt einer Präsentation durchsucht, dabei eine Liste aller vorkommenden Wörter und deren Häufigkeit erstellt und ausgibt.

   (Hinweis: Zur Implementierung kann die in OCaml im Modul `Hashtbl` vordefinierte Hashtabellen-Datenstruktur benutzt werden.)