Informatik I

PD Dr. J.-G. Smaus
G. Röger, R. Mattmüller
Wintersemester 2010/2011

Universität Freiburg
Institut für Informatik

Übungsblatt 7

Abgabe: 10. Februar 2011

Dieses Blatt wird nicht mehr in den Übungen besprochen. Korrigierte Blätter können voraussichtlich ab dem 14. Februar bei Robert Mattmüller (Raum 052-00-045) oder Gabi Röger (Raum 052-00-041) abgeholt werden.

Aufgabe 7.1 (Schleifeninvarianten, 3+3+4 Punkte, Abgabe: auf Papier)

Eine *Schleifeninvariante* ist eine Eigenschaft, die zu einem bestimmten Punkt in jedem Schleifendurchlauf gültig ist (hier: am Ende jedes Schleifendurchlaufs).

Sie können die Gültigkeit einer Schleifeninvariante induktiv über die Anzahl der Schleifendurchläufe beweisen, indem sie zeigen, dass die Invariante zu Beginn des ersten Schleifendurchlaufs gilt, und dass sie, wenn sie am Ende des i-ten Schleifendurchlaufs gilt (und damit zu Beginn des $i+1$-ten), auch am Ende des $i+1$-ten Schleifendurchlaufs gelten muss.

(a) Zeigen Sie, dass für die `while`-Schleife der Funktion `counter` die Schleifeninvariante $\texttt{x} \leq 10$ gilt, unter der Annahme, dass `counter` für eine natürliche Zahl $\texttt{x} < 10$ aufgerufen wird.

```
def counter(x):
    while x < 10:
        x += 1
    return x
```

(b) Zeigen Sie, dass für die `while`-Schleife der Funktion `prod` die Schleifeninvariante $\texttt{result} = \prod_{j=0}^{\texttt{index}-1} \texttt{lst[}j\texttt{]}$ gilt, wenn `prod` mit einer Liste von Zahlen aufgerufen wird. Es gilt die Konvention $\prod_{j=0}^{-1} x_j = 1$.

```
def prod(lst):
    result = 1
    index = 0
    while index < len(lst):
        result *= lst[index]
        index += 1
    return result
```

(c) Zeigen Sie, dass für die `while`-Schleife der Funktion `search` die Schleifeninvariante $\texttt{elem} \in \{\texttt{lst[lower]}, \ldots, \texttt{lst[upper-1]}\}$ gilt, wenn `search` mit einem Element `elem` und einer Liste `lst` von Zahlen aufgerufen wird, wobei `lst` aufsteigend sortiert ist und den Eintrag `elem` enthält.

```
def search(elem, lst):
    lower = 0
    upper = len(lst)
    while lower < upper:
        mid = (upper+lower)//2
        midval = lst[mid]
        if midval == elem:
            return mid
        elif midval < elem:
            lower = mid+1
        else:
            upper = mid
    return -1
```

Aufgabe 7.2 (Doppelt verkettete Listen, 4+2+4 Punkte, Abgabe: in einer Datei `doubly_linked_list.py`)

Eine *doppelt verkettete Liste*[1] ist eine Datenstruktur, die wie eine (einfach) verkettete Liste aufgebaut ist, jedoch zusätzlich zu den `next`-Referenzen auf das folgende Element noch `prev`-Referenzen auf das jeweils vorhergehende Element besitzt.

(a) Implementieren Sie analog zu den einfach verketteten Listen aus dem Modul `linkedlist.py` nun doppelt verkettete Listen. Es bietet sich an, eine Klasse `DoublyLinkedListNode` mit Feldern `data`, `next` und `prev` und eine weitere Klasse `DoublyLinkedList` mit zwei Feldern `first_node` und `last_node` zu implementieren. Folgende Methoden von `DoublyLinkedList` werden verlangt, weitere sind optional:

- `__init__(self)`,
- `is_empty(self)`,
- `first(self)`,
- `last(self)`,
- `insert_after(self, node, newnode)`,
- `insert_before(self, node, newnode)`,
- `insert_beginning(self, newnode)`,
- `insert_end(self, newnode)`,
- `length(self)`.

Wo es sinnvoll ist, wird immer die aktuelle Liste modifiziert, keine neue erzeugt. Hilfsmethoden sind erlaubt und erwünscht.

(b) Implementieren Sie eine Klasse `Bookmark`, die Browser-Lesezeichen repräsentiert. Ein `Bookmark` soll aus einer Beschreibung und einer URL bestehen.

[1] `http://en.wikipedia.org/wiki/Doubly-linked_list`

(c) Implementieren Sie eine Klasse `Favorites`, die eine Liste von Browser-Lesezeichen repräsentiert. Die Einträge sollen durch eine doppelt verkettete Liste von `Bookmark`s dargestellt werden, und die Favoriten-Liste soll in der Lage sein, ihre Einträge aus einer Textdatei zu lesen (das genaue Format, in dem die Einträge in der Datei gespeichert werden, ist Ihnen überlassen). Das *Schreiben* der Einträge in eine Datei benötigen wir hier nicht, Sie können es aber natürlich trotzdem implementieren.

Die Dokumentation der für Datei- und Stringoperationen notwendigen Module, Klassen und Methoden finden Sie auf der Python-Website.[2][3]

(d) *Bonusaufgabe (keine Punkte).* Implementieren Sie einen `FavoritesViewer`, der eine `Favorites`-Liste aus einer Datei einliest und dem Nutzer anzeigt. Der Nutzer soll über die Tastatur in der Liste navigieren können (in beide Richtungen – deswegen die *doppelt* verkettete Liste). Wählt er einen Eintrag aus, so soll die entsprechende URL im Standard-Webbrowser geöffnet werden.

Die Steuerung des Webbrowsers nimmt Ihnen das `webbrowser`-Modul ab.[4] Hinweis: Das `webbrowser`-Modul liefert einen Fehler, wenn die übergebene URL von Whitespace umgeben ist. Sie sollten also beim Einlesen der URLs Whitespace mit der String-Methode `strip()` entfernen.

Die Übungsblätter müssen individuell bearbeitet werden. Gruppenabgaben sind nicht zulässig.

[2] `http://docs.python.org/py3k/tutorial/inputoutput.html#reading-and-writing-files`

[3] `http://docs.python.org/py3k/library/stdtypes.html#string-methods`

[4] `http://docs.python.org/py3k/library/webbrowser.html`