Prof. Dr. Leif Kobbelt
Stefan Dollase, Ira Fesefeldt, Alexandra Heuschling, Gregor Kobsik

Übung 1

Tutoraufgabe 1 (Vollständige und Strukturelle Induktion):

a) In dieser Aufgabe wiederholen wir vollständige Induktion. Beweisen Sie hierfür mittels vollständiger Induktion über n, dass

$$\sum_{i=1}^{n} i \quad = \quad \frac{n \cdot (n+1)}{2} \; .$$

b) Wenn wir Beweise für Datenstrukturen aufschreiben wollen, ist es häufiger eleganter die *Struktur* einer Datenstruktur ausnutzen. Dafür beweisen wir einen entsprechenden Satz mithilfe der *Strukturellen Induktion*. Nehmen wir beispielsweise die Struktur einer Liste, so haben wir die Prädikate

- CREATE(),
- INSERT(a, L), und
- INSERT*(a, L).

Für eine Strukturelle Induktion müssen wir dafür zunächst als Induktions Anfang den Satz für alle atomaren Strukturelemente beweisen, hier also für CREATE(). Danach etablieren wir die Induktions Hypothese und nehmen an, dass der Satz für eine beliebige aber feste Liste L gilt. Anschließend beweisen wir den Satz auch für nicht-atomare Strukturelemente, hier also für INSERT(a, L) und INSERT*(a, L), mithilfe der Induktions Hypothese.

Wir definieren die Funktion LENGTH wie folgt:

$$\begin{aligned} &\text{LENGTH(CREATE())} &&= 0 \\ &\text{LENGTH(INSERT}(a, L)) &&= 1 + \text{LENGTH}(L) \\ &\text{LENGTH(INSERT*}(a, L)) &&= 1 + \text{LENGTH}(L) \end{aligned}$$

Zusätzlich nutzen wir die aus der Vorlesung bekannte Funktion JOIN, welche wie folgt definiert ist:

$$\begin{aligned} &\text{JOIN(CREATE(), } z) &&= z \\ &\text{JOIN(INSERT}(x, y), z) &&= \text{INSERT*(x, JOIN}(y, z)) \\ &\text{JOIN(INSERT*}(x, y), z) &&= \text{INSERT*(x, JOIN}(y, z)) \end{aligned}$$

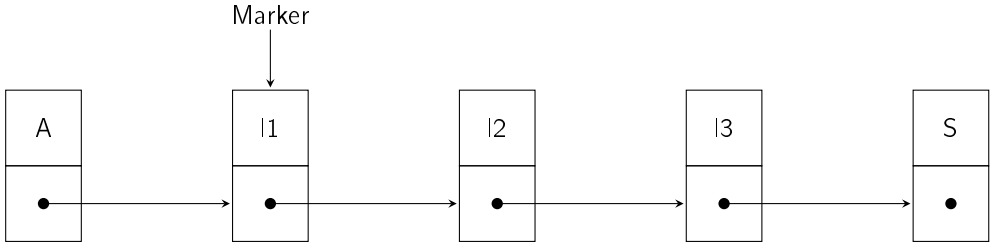
Beweisen Sie nun, dass für alle Listen L_1 und L_2 folgender Satz gilt:

$$\text{LENGTH(JOIN}(L_1, L_2)) \quad = \quad \text{LENGTH}(L_1) + \text{LENGTH}(L_2)$$

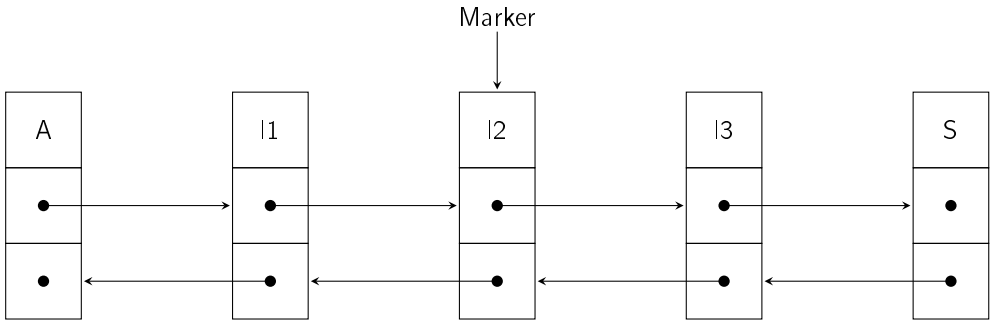
Tutoraufgabe 2 (Listen Operationen):

Wir stellen in dieser Aufgabe Listen, welche mittels Manipulation von Zeigern implementiert werden, wie unten dargestellt graphisch dar. Dabei signalisieren die Punkte · ein Zeiger, der Pfeil zu welchem Objekt der Zeiger zeigt und die Symbole über den Punkten welchen Inhalt das Listenelement hat. Das Anchor Element hat das Symbol A, das Sentinel Element hat das Symbol S. Der Marker Zeiger wird als eigenständiger, markierter Pfeil oben drüber illustriert.

a) Führen Sie erst eine Insert-Operation mit dem Element I4, dann eine Next-Operation und schließlich eine Delete-Operation durch. Geben Sie die Liste nach jeder Operation graphisch an.



b) Führen Sie erst eine Next-Operation, dann eine Delete-Operation, dann eine Previous-Operation und schließlich eine Insert-Operation mit dem Element I4. Geben Sie die Liste nach jeder Operation graphisch an.



Aufgabe 3 (Vollständige und Strukturelle Induktion): 5 + 6 = 11 Punkte

a) In dieser Aufgabe wiederholen wir vollständige Induktion. Beweisen Sie hierfür mittels vollständiger Induktion über n, dass

$$\sum_{i=1}^{n} i^2 = \frac{n \cdot (n+1) \cdot (2 \cdot n+1)}{6} .$$

b) Wenn wir Beweise für Datenstrukturen aufschreiben wollen, ist es häufiger eleganter die *Struktur* einer Datenstruktur ausnutzen. Dafür beweisen wir einen entsprechenden Satz mithilfe der *Strukturellen Induktion*. Nehmen wir beispielsweise die Struktur einer Liste, so haben wir die Prädikate

- CREATE(),
- INSERT(a, L), und
- INSERT*(a, L).

Für eine Strukturelle Induktion müssen wir dafür zunächst als Induktions Anfang den Satz für alle atomaren Strukturelemente beweisen, hier also für CREATE(). Danach etablieren wir die Induktions Hypothese und nehmen an, dass der Satz für eine beliebige aber feste Liste L gilt. Anschließend beweisen wir den Satz auch für nicht-atomare Strukturelemente, hier also für INSERT(a, L) und INSERT*(a, L), mithilfe der Induktions Hypothese.

Wir definieren die Funktion LENGTH wie folgt:

LENGTH(CREATE())	=	0
LENGTH(OVERWRITE(a, CREATE()))	=	0
LENGTH(INSERT(a, L))	=	1 + LENGTH(L)
LENGTH(INSERT*(a, L))	=	1 + LENGTH(L)

Zusätzlich nutzen wir die aus der Vorlesung bekannte Funktion OVERWRITE, welche wie folgt definiert ist:

OVERWRITE(y, INSERT(x, z))	=	INSERT(y, z)
OVERWRITE(y, INSERT*(x, z))	=	INSERT*(x, OVERWRITE(y, z))

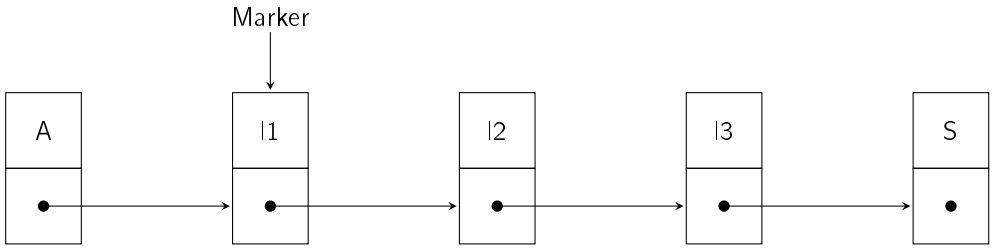
Beweisen Sie nun, dass für alle Listeneinträge a und alle Listen L folgender Satz gilt:

LENGTH(OVERWRITE(a, L)) = LENGTH(L)

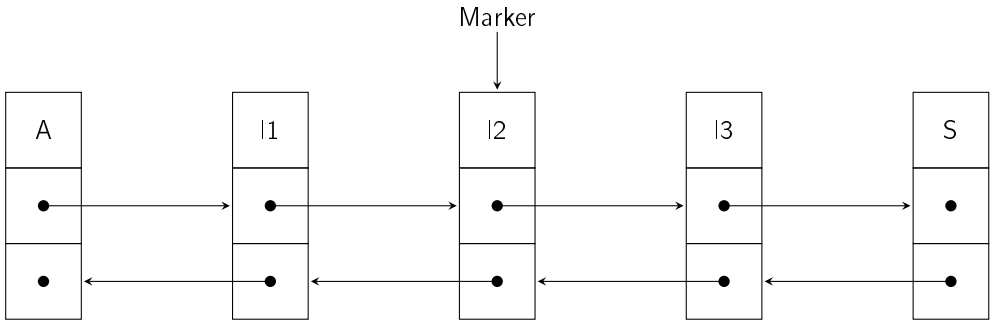
Aufgabe 4 (Listen Operationen): 3 + 6 = 9 Punkte

Wir stellen in dieser Aufgabe Listen, welche mittels Manipulation von Zeigern implementiert werden, wie unten dargestellt graphisch dar. Dabei signalisieren die Punkte · ein Zeiger, der Pfeil zu welchem Objekt der Zeiger zeigt und die Symbole über den Punkten welchen Inhalt das Listenelement hat. Das Anchor Element hat das Symbol A, das Sentinel Element hat das Symbol S. Der Marker Zeiger wird als eigenständiger, markierter Pfeil oben drüber illustriert.

a) Führen Sie erst eine Delete-Operation, dann eine Next-Operation und schließlich eine Insert-Operation mit dem Element I4 durch. Gebe die Liste nach jeder Operation graphisch an.



b) Führen Sie erst eine Previous-Operation, dann eine Insert-Operation mit dem Element l4, dann eine Next-Operation und schließlich eine Delete-Operation. Gebe die Liste nach jeder Operation graphisch an.



Aufgabe 5 (Programmierung in Python: Listen): 2 + 18 = 20 Punkte

Bearbeiten Sie die Python Programmieraufgaben. In dieser praktischen Aufgabe werden Sie lineare Strukturen (einfach und doppelt verkettete Listen) implementieren. Diese Aufgabe dient dazu einige Konzepte der Vorlesung zu wiederholen.
Zum Bearbeiten der Programmieraufgabe können Sie einfach den Anweisungen des Notebooks *blatt01-python.ipynb* folgen. Das Notebook steht in der .zip-Datei zum Übungsblatt im Lernraum zur Vergügung.
Ihre Implementierung soll immer nach dem *# YOUR CODE HERE* Statement kommen.
Laden Sie spätestens bis zur Deadline dieses Übungsblatts auch Ihre Lösung der Programmieraufgabe im Lernraum hoch. Die Lösung des Übungsblatts und die Lösung der Programmieraufgabe muss im Lernraum an derselben Stelle hochgeladen werden. Die Lösung des Übungsblatts muss dazu als .pdf-Datei hochgeladen werden. Die Lösung der Programmieraufgabe muss als .ipynb-Datei hochgeladen werden.

Übersicht der zu bearbeitenden Aufgaben:

a) Einfach verkettete Listen

- insert()
- delete()

b) Doppelt verkettete Listen

- insert()
- delete()
- length()
- join()
- split()
- cycle()
- swap()
- reverse()