Übungen zu **Algorithmen und Programmierung 3** - Blatt 1- H. Schweppe / M. Walter WS 95/96

Abgabe: Do. 26.10.95 bis 12.00

Diese Übung dient im wesentlichen dazu, Stoff aus Alp 2 aufzufrischen.

*** Aufgabe 1 (15 P)**
Erweitere das Modul `LinkedLi`, das eine einfach verkettete Liste realisiert, um die Operation :
[
```
        PROCEDURE Reverse;
        (* Reverse [] = [] *)
        (* Reverse [x1,...xn-1, xn] = xn: Reverse [x1..xn-1]*)
```

a) (7 P) Entwickele eine iterative Lösung, ohne die Liste zu kopieren und ohne NEW zu verwenden.
b) (3 P) Entwickele eine rekursive Lösung `RevRec`.
c) (2 P) Arbeitet die Prozedur `Reverse` korrekt, wenn der Listenkopf als Wertparameter übergeben wird? (Begründung)
d) (3 P) Ergänze die Implementierung der Prozedur `Map(p: Proc)`, die p auf jedes Element der Liste anwendet.

Ein Definitionsmodul, eine teilweise Implementierung (`linkedli.def/mod`), ein Typmodule (`tmodule.def/mod`) und eine Testumgebung (`tstlili.mod`) stehen Ihnen zur Verfügung (Verzeichnis `~walter/alp3`)

*** Aufgabe 2 (11 P)**
a) (3 P) Die Vorbedingung P (man kann auch sagen: Voraussetzung, *precondition*) eines Programms S soll möglichst schwach sein, die Nachbedingung Q (auch: der Effekt, *postcondition*) möglichst stark. Die folgenden Deduktionsregeln sind nützlich, um bei möglichst schwachen Voraussetzungen möglichst starke Effekte zu haben:

wenn $P \{S\} Q$ *und* $P' \rightarrow P$ dann $P' \{S\} Q$
wenn $P \{S\} Q$ *und* $Q \rightarrow Q'$ dann $P \{S\} Q'$

ferner gelten:

wenn $P \{S\} Q$ *und* $P \{S\} Q'$ dann $P \{S\} Q \wedge Q'$
wenn $P \{S\} Q$ *und* $P' \{S\} Q$ dann $P \vee P' \{S\} Q$

Beweisen Sie:
Wenn gilt: $P \{S\} Q$ und $P' \{S\} Q'$ dann $P \wedge P' \{S\} Q \wedge Q'$ und $P \vee P' \{S\} Q \vee Q'$

b) (7 P) Das folgende Programmstück ist durch die gegebenen Vor- und Nachbedingungen P und Q spezifiziert.
```
        VAR a,b: INTEGER;
        ...
        (* P: ( b >= -3 ) AND ( b <= 0 ) *)
        a:= -b;
        IF a >= 6 THEN b:= a+b ELSE b := a - b END;
        (* Q: (b > 0) AND (b <= 4 )
```
a) Bestimmen Sie die schwächste Vorbedingung zu Q. (Hinweis: Bestimmen Sie erst die schwächste Vorbedingung P1 zur Fallunterscheidung, die Q garantiert, dann die schwächste Vorbedingung P0 zur Zuweisung, die P1 garantiert)
b) Ist das Programmstück korrekt? (Beweis)

Aufgabe 3 (4 P)
Wo liegt der Fehler im folgenden „Induktionsbeweis"? Korrigiere den Fehler!

Beh.: $\sum_{k=1}^{n} k = O(n)$

Ind.vorauss. $\sum_{k=1}^{1} k = O(1)$

Ind.schluß: $\sum_{k=1}^{n+1} k = O(n)+(n+1) = O(n+1) = O(n)$

Hinweise:
Übungsblätter werden ein- bis zweiwöchentlich ausgegeben. Von n Blättern müssen n-1 bearbeitet werden. Programmieraufgaben können in Zweiergruppen bearbeitet werden, theoretische (d.h. keine Implementierung) sollen allein gelöst und abgegeben werden. Die mit * gekennzeichneten Aufgaben müssen bearbeitet werden.
Jeder muß mindestens einmal im Semester seine Lösung im Tutorium vortragen. Die Lösungen auf Papier müssen zum angegebenen Termin in der VL abgeliefert werden. Programmieraufgaben werden in folgender Form abgegeben:

(i) Papierversion des Quellprogramms, natürlich mit Spezifikation und Kommentierung
(ii) `<prg>.mod`, `<prg>.exe` und die beiden Dateien mit Testdaten und Protokoll (s.u.) müssen zum Abgabetermin in das Verzeichnis des Tutors `~kilian/alp3` bzw. `~schaefer/alp3` (Hr. Meder!) kopiert sein.
`<prg>    ::= t<tutorium>g<gruppennr>a<blatt-aufgabe>`
`<tutorium>::= 1 | 2 | 3 | 4` (* Tutorien in zeitlicher Reihenfolge durchnumeriert*)
(usw., siehe folgendes Beispiel)

Jede **Implementierung muß getestet werden**. Entwerfen Sie dazu eine Datei mit Testdaten und erzeugen Sie damit eine Protokolldatei mit Testergebnissen. Das kann man sehr einfach durch Umlenkung von Eingabe bzw. Ausgabe machen:

```
2g3a1-2.exe > 2g3a1-2.log < 2g3a1-2.tst
```

wobei `2g3a1-2.exe` das ausführbare Programm, `2g3a1-2.tst` bzw. `2g3a1-2.log` die Testdaten bzw. -ergebnis-Datei der Gruppe 3 aus Tutorium 2 (Mo. 16 - 18) für Blatt 1, Aufgabe 2 sind. `2g3a1-2.tst` bzw. `2g3a1-2.log` sind Bestandteil einer Lösung!

Programmiert wird in Modula-2 an den **PCs des Instituts,** außerdem in Miranda auf den Suns (Gofer tut's auch). Leider sind einige der Pentium-Rechner gestohlen worden (halten Sie im eigenen Interesse die Augen auf - und den Dieb), deshalb muß evtl. auf die schon leicht angestaubten 286er-PCs ausgewichen werden. Wir verwenden TopSpeed-Modula2 Vers. 3.02. Wegen der objektorientierten Erweiterungen gibt es dazu keine Alternative!

Am Dienstag, **13.2.96** wird eine 4-stündige **Klausur** geschrieben (8 - 12).
Für den **Übungsschein** sind **mindestens 50 %** der Gesamtpunktzahl der n Übungsblätter *und* 50 % der Klausurpunkte erforderlich, ferner mindestens eine Lösungspräsentation im Tutorium.

Sprechstunden: siehe Aushang und WWW-Seite