# Universität Karlsruhe

## Informatik 1 WS 00/01

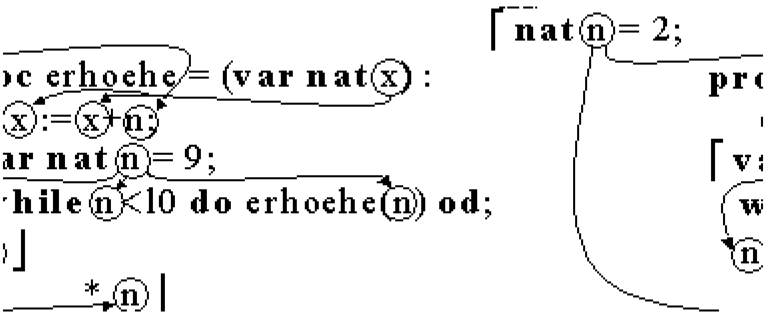
Institut für Telematik, Forschungsgruppe C&M
Prof. Dr. S. Abeck, R. Scholderer

# Musterlösungen zum Übungsblatt 14

## Lösung

### 1. Gültigkeit und Lebensdauer (II)

**Teilaufgabe 1:**




**Teilaufgabe 2:**
Es sei n* der Identifikator im inneren Block des Programmabschnitts.

| | Lebensdauer | | | Gültigkeit | | |
|---|---|---|---|---|---|---|
| | n | x | n* | n | x | n* |
| ⌈ **nat** n = 2 | • | | | • | | |
| proc erhoehe = (var nat x) : | • | • | | • | • | |
| x:= x+n; | • | • | | • | • | |
| ⌈ **var nat** n = 9: | • | | • | | | • |
| **while** n < l0 **do** erhoehe(n) | • | | • | | | • |
| n ⌋ | • | | • | | | • |
| * n ⌋ | • | | | • | | |

**Teilaufgabe 3:**
Statische Bindung: Das Ergebnis des Abschnitts ist erhoehe(9) * 2 = (2+9) * 2 = 22.
Dynamische Bindung: Das Ergebnis des Abschnitts ist erhoehe(9) * 2 = (9 + 9) * 2 = 36.

---

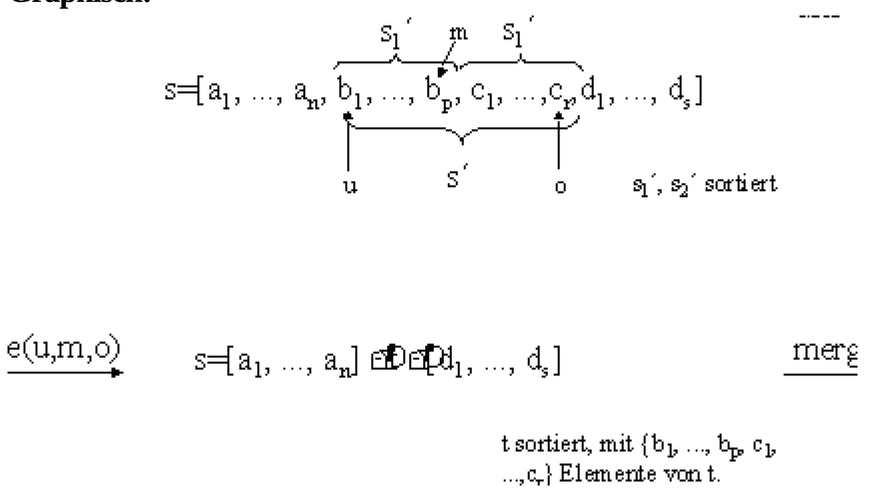
## Lösung

## 2. Rekursive Prozeduren: Aufrufbaum

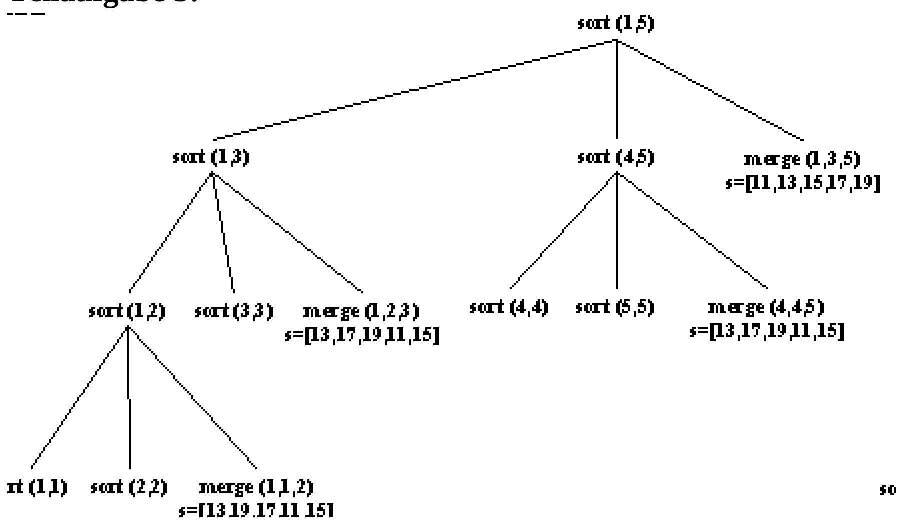
**Teilaufgabe 1:**
s = [19, 13, 17, 11, 15]

**Teilaufgabe 2:**
Parameter von merge:
Die merge übergebenen Paramter markieren Anfang, Mitte und Ende einer s´ Teilsequenz von s.
u: Index des ersten Elementes von s´.
o: Index des letzten Elements von s´.
m: Index des mittleren Elements von s´ (Wird berechnet als (u+o)÷2).
Funktion von merge:
Die Teilsequenz s´ besteht aus zwei sortierten Teillisten s1´ und s2´: s1´ geht von u bis m, s2´ von m+1 bis o (u, m, o sind Indizes der jeweiligen Elemente in s).
Merge erzeugt nun aus s1´ und s2´eine einzige sortierte Liste t (bestehend aus den Elementen von s1´ und s2´). Anschließend wird s´ in s durch t ersetzt.

**Graphisch:**




**Teilaufgabe 3:**




---

# Lösung

## 3. Zusicherungsmethode

a)

| t | n | |
|---|---|---|
| [7, 3, 6, 4, 3, 1] | 0 | vor 1. Schleifendurchlauf |
| [3, 6, 4, 3, 1] | 0 | nach 1. Schleifendurchlauf |
| [6, 4, 3, 1] | 1 | nach 2. Schleifendurchlauf |
| [4, 3, 1] | 1 | nach 3. Schleifendurchlauf |
| [3, 1] | 1 | nach 4. Schleifendurchlauf |
| [1] | 2 | nach 5. Schleifendurchlauf |
| [] | 2 | nach 6. Schleifendurchlauf |

Funktion von S: In der Variable n wird gezählt, wie viele Listenelemente mit dem Wert a in der Sequenz l auftreten.

b)
Bevor die Schleife erstmals durchlaufen wurde, gilt t = l aufgrund der Zuweisung zu Beginn. Mir r = [] ergibt sich conc(r, t) = conc ([], l) = l.
In jedem Schleifendurchlauf wird jeweils das erste Element von t entfernt. t ist also stets eine Teilsequenz von l, wobei nach dem i-ten Schleifendurchlauf das k-te Element von t mit dem i+k-ten (k = 1,2,3,...; i = 1,2,3,... ) Element von l übereinstimmt.

Es seien also $l = [l_1, ..., l_i, l_{i+1},...,l_n]$ und $t = [t_1, ...,t_m]$, dann gilt: $t = [t_1, ...,t_m] = [l_{i+1},...,l_n]$. Wählt man also für r eine Teilliste von l, die vom 1. bis zum i-ten Element von l geht, also $r = [l_1, ..., l_i]$, so gilt wieder conc(r,t) = $conc([l_1, ..., l_i], [l_{i+1},...,l_n]) = [l_1, ..., l_i, l_{i+1},...,l_n] = l$. Nach Verlassen der Schleife ist t = []. Mit $r=[[l_1, ..., l_i] = l$ gilt wieder die Invariante, da $conc(r,[]) = conc([l_1, ..., l_i],[]) = l$.

c) Die Zusicherungen werden wie folgt ergänzt:

| | | |
|---|---|---|
| (1) | | {true} |
| (2) | **var seq m** t := l; | |
| (3) | | {t = l} |
| (4) | **var m** x := a; | |
| (5) | | $\{t = l \wedge x = a\}$ |
| (6) | **var nat** n : = 0: | |
| (7) | | $\{t = l \wedge x = a \wedge n = 0\}$ |
| (8) | | $\{\exists r: l = conc(r,t) \wedge n = \#_a (r) \wedge x = a\}$ |
| (9) | **while** | |
| (10) | ¬isempty(t) | |
| (11) | **do** | |
| (12) | | $\{\neg isempty(t) \wedge \exists r: l = conc(r,t) \wedge n = \#a (r) \wedge x = a\}$ |
| (13) | **if** | |

| | | |
|---|---|---|
| (14) | $first(t) \stackrel{?}{=} x$ | |
| (15) | **then** | |
| (16) | | $first(t) = x \land \neg isempty(t) \land \land r{:}\ l = conc(r,t) \land n = \#x\ (r) \land x = a \}$ |
| (17) | | $\{\exists r{:}\ l = conc(r, rest(t)) \land n+1 = \#_a (r) \land x = a \}$ |
| (18) | n := n+1 | |
| (19) | | $\{\exists r{:}\ l = conc(r, rest(t)) \land n = \#a\ (r) \land x = a \}$ |
| (20) | **else** | |
| (21) | | $\{first(t) \neq x \land \neg isempty(t) \land \land r{:}\ l = conc(r,t) \land n = \#x\ (r) \land x = a \}$ |
| (22) | | $\{\exists r{:}\ l = conc(r, rest(t)) \land n = \#a\ (r) \land x = a \}$ |
| (23) | **nop** | |
| (24) | | $\{\exists r{:}\ l = conc(r, rest(t)) \land n = \#_a (r) \land x = a \}$ |
| (25) | **fi;** | |
| (26) | | $\{\exists r{:}\ l = conc(r, rest(t)) \land n = \#_a (r) \land x = a \}$ |
| (27) | t := rest(t); | |
| (28) | | $\{\exists r{:}\ l = conc(r,t) \land n = \#a\ (r) \land x = a \}$ |
| (29) | **od** | |
| (30) | | $\{isempty(t) \land \exists r{:}\ l = conc(r, rest(t)) \land n = \#_a (r) \land x = a \}$ |
| (31) | | $\{n = \#_a(1)\}$ |