4./5. Übung zur *Nichtsequentiellen Programmierung* (Ch. Maurer, WS 97/98)

1. Arbeiten Sie sich durch die Lösung des “zweiten” Leser-Schreiber-Problems (Priorität für die Schreiber) von Courtois, Heymans und Parnas (Comm. ACM 14 (1971), 667–668); zu finden auch bei Andrews, Ex. 4.20 (S. 219), Burns-Davies 6.8.1 (S. 187) oder Herrtwich-Hommel, 11.3.3 (S. 212 in der alten Auflage).

2. Mitten durch die Eiger-Nordwand zieht sich eine einspurige Straße mit bröckelnder äußerer Fahrbahnkante ohne Leitplanken und Wendestellen.

Allein von Ihrer Fähigkeit, mit Semaphoren das derzeitige heftige Verkehrsaufkommen in beiden Richtungen von Weihnachtsmännern auf achtzehnspännigen Elchfuhrwerken im Schneegestöber auf dieser vereisten Straße zu regeln, hängt deren Sicherheit ab.

3. Additive Semaphore sind durch die Operationen

$$\texttt{P(S,n)} = \langle \texttt{Await S >= n; S:= S - n} \rangle \quad \text{und} \quad \texttt{V(S,n)} = \langle \texttt{S:= S + n} \rangle$$

definiert (allgemeine Semaphore sind ein Spezialfall für `n = 1`).

a) Konstruieren Sie additive Semaphore unter Rückgriff auf allgemeine Semaphore.

b) Verwenden Sie Ihre Konstruktion zur Lösung des Leser-Schreiber-Problems.

4. Lösen Sie mit Semaphoren das klassische Problem der “speisenden Philosophen”.

Studieren Sie dabei gründlich alle denkbaren Fehlersituationen.

Fakultative Zusatzaufgabe (Abgabetermin freigestellt):

5. Implementieren Sie den in der Vorlesung spezifizierten Modul `PROZESSE` in Modula-2 (empfohlen: MOCKA = **MO**dula-2-**C**ompiler-**KA**rlsruhe) oder in C unter Rückgriff auf geeignete UNIX-Systemaufrufe zur Prozeßsteuerung (`fork`, `wait` und `exit`) und zur Prozeßkommunikation mit Hilfe von Semaphoren (`semget` und `semop`).

Angenehme Weihnachtstage und einen guten Start
in ein gesundes und erfolgreiches Neues Jahr!