Christian-Albrechts-Universität zu Kiel
Institut für Informatik und Praktische Mathematik
Prof. Dr. W.-P. de Roever
Martin Steffen

Informatik III (OS)

Wintersemester 2003/04 — **Serie 2** — 28. Oktober 2003

Thema : Synchronisation

Ausgabetermin: 28. Oktober 2003

Abgabe: 12. November 2003

Zwei Worte vorneweg:

1. die meisten haben es mitbekommen:
   (a) Die Seite mit Informationen zum Kurs (Übungszettel etc.) lautet
   `http://www.informatik.uni-kiel.de/inf/deRoever/WS0304/OS/`
   Wichtige Dinge werden allerdings in der Vorlesung angesagt und ausgeteilt.
   (b) das studentische Forum unter
   `http://www.infmath.de`
   bietet ebenfalls Informationen und Diskussionen zu (unter anderem) dieser Vorlesung.
2. Das Semester fängt immer ein wenig ruckelig an, bis sich der Staub der Übungsgruppen-Zuteilung gelegt hat und sich der Ausgabe-Abgabe-Korrektur-Zyklus eingeschwungen hat (diesmal besonders, da *kommende Woche* die Vorlesung (nicht die Übungen) ausfällt. Wegen der Nachfrage: als Schema streben wir an[1]

   zur Vorlesung: *Ausgabe* der Zettel am Dienstag
   + **ein Woche: Mittwoch** *Abgabe* der Zettel *Mittwochs* in den Schrein in Zweiergruppen, die sich für die Dauer des Semesters zusammenfinden.
   + **eine Woche, Übung** Rückgabe und Besprechung. dazu natürlich Fragen zu vergangenen Vorlesungen sowie Fragen zum neuen Zettel ("wie ist die Aufgabe zu verstehen?" ...)

[1] diese Zettel hält sich noch nicht richtig daran, da kommende Woche die Vorlesung nicht stattfindet.

Aufgabe 1 (Dekker (6 Punkte)) Untersuchen Sie den angegebenen Algorithmus auf folgende Eigenschaften:

1. Gewährleistung des gegenseitigen Ausschlusses.
2. Freiheit von Verklemmung (oder *deadlock*-Freiheit).
3. Kann die Terminierung („Absturz“) eines Prozesses in seinem nicht-kritischen Abschnitt den Partnerprozeß negativ beeinflussen?
4. Verhält sich die Lösung gutwillig, wenn ein Prozeß wesentlich öfter als der andere in seinen kritischen Abschnitt will?
5. Kann es passieren, daß ein Prozeß, obwohl er seinen kritischen Abschnitt betreten will, unendlich lange darauf wartet?

Beweisen Sie jeweils Ihre Antwort.

```
//  Dekkers Algorithmus f"ur Gegenseitigen Ausschlu"s
//  boolean  P0_wants_to_enter, P1_wants_to_enter = false;

  process P0 =
  begin

    while true                          /* Endlosschleife       */
    do
      some_actions;                     /* unkritischer Code    */

      P0_wants_to_enter := true;
      while  P1_wants_to_enter
      do
        if     favored = 1
        then   P0_wants_to_enter := false;
               while favored = 1 do skip od;
               P0_wants_to_enter := true
        fi
      od;
      critical_section_0;
      favored := 1;
      P0_wants_to_enter := false;
    od;
  end;
```

Aufgabe 2 (Produzent/Konsument (3 Punkte)) Ein klassisches Prozeßsynchronisationsproblem ist das *Produzenten/Konsumenten*-Problem.[2] Gegeben sei folgendes Programmfragment, welches 2 Prozesse definiert:

---

[2] Auch als das Problem des beschränkten Puffers bekannt (auf Englisch *producer/consumer*, *bounded-buffer*).

```
/* producer/consumer Pseudocode                                    */

#define N   100                     /* Plaetze im Puffer   */
int buffer[N];
int count   0;                      /* belegte Plaetze     */

...

producer ()
{
  while (TRUE) {
    ...
    buffer[count++] = produce_item();
    ...
  }
}

consumer ()
{
  while (TRUE) {
    ...
    consume_item(buffer[count--]);
    ...
  }

}
```

Die Absicht ist, die notwendige *Synchronisation* zwischen Konsument und Produzent, um einen Zugriff auf den Buffer außerhalb der Arraygrenzen zu verhindern, durch Verwendung von Semaphoren zu bewerkstelligen. Ergänzen Sie den Code entsprechend und begründen Sie die Korrektheit Ihrer Lösung!