Lösungsvorschlag Aufgabe 5:

a) Was ist die Grundidee bei (Φ, Θ) - Synchronisationssystemen ?

vgl. F. Hofmann:
Betriebssysteme: Grundkonzepte und Modellvorstellungen, 1991: S. 70 ff

Die Überprüfung von Ausführbarkeitsbedingungen verlangt hohen dynamischen Aufwand. Die Grundidee liegt darin, nur dann eine Überprüfung der Ausführbarkeitsbedingung vorzunehmen, wenn sich wenigstens eine der darin vorkommenden Variablen seit der letzten Überprüfung geändert hat.

Erreicht wird dies dadurch, daß

- die Zustandsmenge in zwei Teile aufgespalten wird:
  + Semaphore und
  + 'normale' Variablen.

  In den Ausführbarkeitsbedingungen darf nur auf die Semaphore und den Befehlszähler Bezug genommen werden, nicht auf die 'normalen' Variablen.

- die Aktionen unterteilt werden in synchronisierende und nichtsynchronisierende. Die synchronisierenden Aktionen arbeiten nur mit den Semaphoren und nicht mit den anderen Variablen. Bei ihnen sind keine Sprünge im Programmablauf möglich.
  Die nichtsynchronisierenden Aktionen dürfen nicht auf den Semaphoren arbeiten, sondern nur die 'normalen' Variablen lesen und verändern. Hier sind Sprungbefehle erlaubt.

Da synchronisierende Aktionen im Vergleich zu nichtsynchronisierenden Aktionen vergleichsweise sehr selten sind, kann der dynamische Aufwand erheblich reduziert werden, denn es müssen nur noch sehr selten die Ausführbarkeitsbedingungen überprüft werden.

b) Wie können allgemeine NBL-Bedingungen bzw. beliebige Synchronisationssysteme ohne aktives Warten auf Monoprozessoren mit binären Semaphoren implementiert werden?

vgl. F. Hofmann:
Betriebssysteme: Grundkonzepte und Modellvorstellungen, 1991: S. 67 ff, 70ff, 89ff

Aktion mit NBL-Bedingung bei Prozeß i:

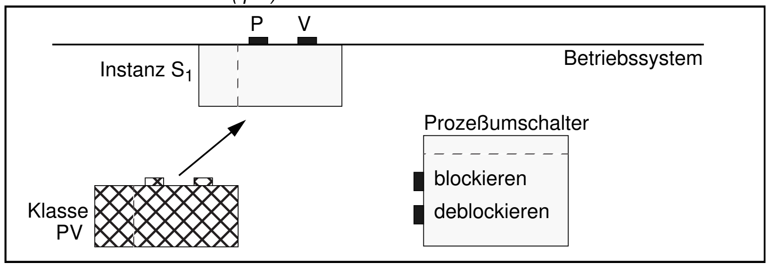
```
while ( TRUE ) {
   P1(mutex);
   if ( NBL ) {
            EFFECTS;

            for k := 1 to n
                     if ( okk = blockiert)
                              V1(Sk);
                     okk := unsynchronisiert;
            V1(mutex);
            break;
   } else {
            oki := blockiert;
            V1(mutex);
            P1(Si);
   }
}
```

Erläutern Sie kurz das Prinzip!

Alle Aktionen müssen unter gegenseitigem Ausschluß laufen.

Für jeden Prozeß i existiert ein privater Semaphor S_i, an dem er sich blockiert, wenn seine Nichtblockierungsbedingung nicht erfüllt ist. Nach einer EFFECTS-Anweisung muß jeder blockierte Prozeß seine Nichtblockierungsbedingung erneut überprüfen.

c) Sie sind Betriebssystementwickler.
Sie sollen an der Betriebssystemschnittstelle die Funktionalität eines PV -Systems bereitstellen. Das Betriebssystem sei aus Modulen aufgebaut.
Es existiert bereits ein Modul **Prozeßumschalter** mit den Operationen
blockieren(pid) und
deblockieren(pid).



Unter welchen Voraussetzungen ist eine Lösung möglich?

Alle Teile im Betriebssystemkern müssen unter gegenseitigem Ausschluß ablaufen.
Es dürfen maximal MAX_INTEGER Prozesse gleichzeitig aktiv sein.

Skizzieren Sie eine Lösung für das Modul **PV**!

Implementierungsskizze:

```
zaehler:        INTEGER              = startwert;
warteschlange:  SEQUENCE OF PID      = leer;

P()
        zaehler := 'zaehler - 1;

        if ( zaehler < 0 ) then
                push( warteschlange, meine_pid );
                prozessumschalter->blockieren( meine_pid);
        endif

V()
        zaehler := 'zaehler + 1;

        if ( zaehler <= 0 ) then
                prozessumschalter->deblockieren( pop( warteschlange ));
        endif
```

Der Zähler wird, wenn er negativ ist, zum Zählen der wartenden Prozesse benutzt.
In dem Beispiel ist nichts über die Strategie beim push oder pop auf die Warteschlange ausgesagt.

d) Sie sind Anwendungsprogrammierer.
An der Betriebssystemschnittstelle steht die Funktionalität des Moduls **Prozeßumschalter** mit den Operationen *blockieren(pid)* und *deblockieren(pid)* zur Verfügung.
Ist es Ihnen möglich das Modul **PV** zu implementieren?

Nein.
Es gibt keine Operationen, um den gegenseitigen Ausschluß zu gewährleisten.

Lösungsvorschlag Aufgabe 5b:

Nach dem Invarianztheorem von Lipton ist es eine notwendige (aber nicht hinreichende!!!) Bedingung für die Gültigkeit der Beziehung Prozeßsystem Q simuliert Prozeßsystem P, daß ein von P implizit definierter Σ-Schnitt Π auch von Q implizit definiert wird.

Durch diese Bedingung kann der geforderte Beweis durchgeführt werden, indem ein Σ-Schnitt Π in P identifiziert wird, der in Q nicht definierbar ist. Vereinfacht wird diese Vorgehensweise durch einige Definitionen, in denen die Schnitte, die von einem speziellen Koordinierungssystem definierbar sind, spezifiziert werden. So gilt für PV- und PV_k-Systeme:

Jeder PV- und PV_k-definierbare Ausschlußschnitt ist ein Äquivalenzschnitt. (Zur Definition von Ausschlußschnitt und Äquivalenzschnitt sei auf die Vorlesungsfolien verwiesen).

Ein Ausschlußschnitt des gegebenen Prozeßsystems ist durch w = (1,1,1,1,0,0,0) gegeben mit:

R	σ_1	σ_2	σ_3	σ_4
σ_1	X			
σ_2		X		
σ_3	X	X	X	
σ_4	X	X		X

wobei gilt: $\sigma_1 = a_{1,1}$; $\sigma_2 = a_{2,1}$; $\sigma_3 = a_{3,1}$ und $\sigma_4 = a_{4,1}$.

Da dieser Ausschlußschnitt nicht symetrisch ist, stellt er auch keinen Äquivalenzschnitt dar. Da aber nur Äquivalenzschnitte durch PV- und PV_k-Systeme definierbar sind, kann das gegebene Prozeßsystem durch diese Koordinierungssysteme nicht simuliert werden.