Übung 3

Aufgabe 1 *Prozesssteuerung in Ada mit Hilfe von Rendezvous*

Das "einfache eingebettete System" aus der Vorlesung (S. 2-41 ff) kann durch eine dritte Task ergänzt werden, die für das Entgegennehmen der gemessenen Druck und Temperaturwerte sowie deren Darstellung an der Konsole zuständig ist (vgl. Übung 2, Aufgabe 4).

Realisieren Sie das Package `IO` und die Task `Console` so, dass die sie mit den `Controller`-Tasks über ein Rendezvous kommunizieren.

Aufgabe 2 *Zufahrtskontrolle zu einem Parkplatz*

Lösen Sie das Parkplatzproblem (Vgl. Übung 1, Aufgabe 5 und Übung 2, Aufgabe 5) mit Hilfe dreier nebenläufiger Tasks: eine steuert die Einfahrtsschranke, eine steuert die Ausfahrtsschranke und eine steuert das Signal.

Skizzieren Sie eine Lösung in Ada, bei der sich Die Tasks über Rendezvous synchronisieren.

Hinweis: Im Gegensatz zur entsprechenden Aufgabe in Übung 2 muss die Anzahl der Fahrzeuge auf dem Parkplatz hier keine gemeinsame Ressource sein. Es genügt, wenn die Signal-Task diese Anzahl kennt und das Signal auf Rot setzt, wenn der Parkplatz voll ist.

Aufgabe 3

Eine Server-Task habe die folgende Ada-Spezifikation:

```
task Server is

  entry Service_A;
  entry Service_B;
  entry Service_C;

end Server;
```

Schreiben Sie den Rumpf (body) der Task Server, so dass die folgenden Abläufe gewährleistet sind:

- Solange für alle drei Eintrittspunkte Client-Tasks Rendezvous-bereit sind, sollen diese in einer zyklischen Reihenfolge akzeptiert werden (A, B, C, A, B, C,…)
- Falls es für einige (1 oder 2) der Eintrittspunkte keine Rendezvous-bereiten Client-Tasks gibt, sollen die restlichen Eintrittspunkte (für die es Rendezvous-bereite Client-Tasks gibt) in zyklischer Reihenfolge abgearbeitet werden. Die Server-Task sollte nie blockiert werden, solange es Rendezvous-bereite Tasks gibt.
- Wenn es keine Rendezvous-bereiten Tasks gibt, soll die Server-Task kein "busy-waiting" durchführen, sondern blockiert werden, bis ein Eintrittspunkt gerufen wird.
- Wenn alle möglichen Clients terminiert haben, soll die Server-Task auch terminieren.

Hinweis: Für einen Eintrittspunkt **`E`** kann man mit dem Attribut **`E'count`** die Anzahl der an diesem Eintrittspunkt Rendezvous-bereiten Tasks bestimmen (d.h. die Anzahl der Tasks, die den Eintrittspunkt aufgerufen haben und auf ein `accept` warten).