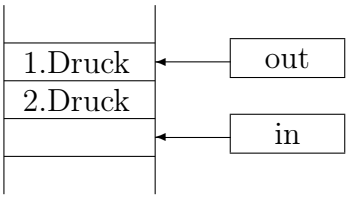
Aufgabe 1: Analysieren Sie die folgende Situation:
Zwei Benutzerprozesse p_1 und p_2, die drucken wollen, sprechen einen Druckerprozeß an, der Druckaufträge über eine Datei annehmen kann, die er nach dem FIFO-Prizip abarbeitet:



p_1:

```
⋮    ⋮
101  lies in;
102  schreibe ‘druck1’, in
103  in := in +1
⋮    ⋮
```

p_2:

```
⋮    ⋮
101  lies in;
102  schreibe ‘druck2’, in;
103  in := in +1
⋮    ⋮
```

Die Prozesse p_1 und p_2 wollen also beide einen Druckauftrag loswerden, indem sie in in die Datei an die Position `in` schreiben.

Der Prozeß könnte nach `101` unterbrochen werden. Beide Prozesse lesen dann denselben Wert für `in` und schreiben ihre Druckaufträge in dieselbe Stelle. Ein Druckauftrag wird dabei überschrieben.

Aufgabe 2: Leser-Schreiber-Problem
Viele Prozesse greifen auf eine gemeinsame Datenstruktur zu:

- beliebig viele Prozesse dürfen gleichzeitig lesen
- ein schreibender Prozeß braucht exklusiven Zugriff auf die Datenstruktur

Lösen Sie dieses Mutual-Exclusion-Problem mithilfe der Ihnen bekannten Synchronisationsprimitive.

mögliche Lösung:

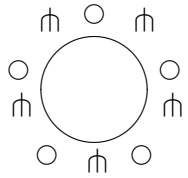
Leser:

```
wiederhole
  P(s)
  lesezähler := lesezähler+1
  wenn lesezähler=1
  dann down(db)
  V(s)
  liesDatenbank
  P(s)
  lesezähler := lesezähler-1
  wenn lesezähler=0
  dann up(db)
  V(s)
  ⋮
ständig
```

Schreiber:

```
wiederhole
  ⋮
  down(db)
  schreibeDatenbank
  up(db)
  ⋮
ständig
```

Aufgabe 3: Problem der Prozeßsynchronisation



Gesucht: eine Lösung des 5-Philosophen-Problems:

- kein Philosoph darf verhungern.
- eine Gabel darf nur von einem Philosophen zugleich benutzt werden.
- nur die Gabeln unmittelbar rechts und links vom Teller dürfen benutzt werden.
- zum Essen braucht man zwei Gabeln.

5-Philosophen-Problem Lösung ohne Verklemmung und ohne Verhungern:
Philosoph:

```
wiederhole
  denken;
  P(s);
  nimm die linke Gabel;
  nimm die rechte Gabel;
  essen;
  lege die linke Gabel hin;
  lege die rechte Gabel hin;
  V(s);
ständig
```

Nachteil: nur ein Philosoph kann essen

Abhilfe:

- eine Semaphore pro Philosoph
- ein Zustand pro Philosoph
  - essend
  - hungrig
  - denkend

```
Philosoph(p) {p = Identifikation des Philosophen}
  wiederhole
       denken;
       nimm_zwei_Gabeln(p); {oder warte}
       essen;
       lege_Gabeln_hin(p);
  ständig;

nimm_zwei_Gabeln(p)
  P(mutex);
  zustand[p] := hungrig;
  versuche(p); {beide Gabeln zu nehmen}
  V(mutex);
  P(s[p]);

versuche(p)
  wenn zustand[p] = hungrig
       und nicht zustand[links(p)] = essend
```

```
      und nicht zustand[rechts(p)] = essend
  dann zustand[p] := essend
      V(s[p])

lege_Gabeln_hin(p)
  P(mutex)
  zustand[p] := denkend;
  versuche(links(p));
  versuche(rechts(p));
  V(mutex)
```

Aufgabe 4: das Bankiersproblem
Der Bankier einer Kleinstadt hat einer Reihe von Kunden ein Kreditvolumen eingeräumt.

Annahme: Nicht alle Kunden werden ihre Kredite gleichzeitig voll ausschöpfen.

Beispiel: Er reserviert \$10.000 für vier Kunden, die insgesamt \$22.000 beanspruchen könnten.

Anfangszustand:

Kunde	hat	max
A	\$0	\$6.000
B	\$0	\$5.000
C	\$0	\$4.000
D	\$0	\$7.000

Gesucht: Verklemmungsfreie Zuteilungsstrategie, wenn Kunden von Zeit zu Zeit Krediterhöhungen nachfragen.

Beispiel: sicherer Zustand

A	\$1.000	\$6.000
B	\$1.000	\$5.000
C	\$2.000	\$4.000
D	\$4.000	\$7.000
	$\sum$ \$8.000	$\sum$ \$22.000

C kann noch Kredit gewährt werden.

Beispiel: unsicherer Zustand

A	\$1.000	\$6.000
B	\$2.000	\$5.000
C	\$2.000	\$4.000
D	\$4.000	\$7.000
	$\sum$ \$9.000	$\sum$ \$22.000

Wie können solche Zustände vermieden werden?

Lösung:

sicherer Zustand

- keine Verklemmung
- alle anstehenden (potentiellen maximalen) Anforderungen können in irgendeiner Reihenfolge befriedigt werden

unsicherer Zustand = nicht sicher

verbale Beschreibung: Der Bankier prüft alle anstehenden Anforderungen, ob die Kreditgewährung in einen unsicheren Zustand führt oder nicht. Um festzustellen, ob ein Zustand sicher ist, ist zu prüfen, ob für einen oder mehrere Kunden deren Maximalforderung erfüllt werden könnte. Wenn das der Fall ist, wird angenommen, daß deren Kredite zurückgezahlt werden. Jetzt wird für die übrigen Kunden (derjenige, der seinem Maximum am nächsten ist, zuerst) nach dem gleichen Verfahren geprüft, ob deren Maximum gewährt werden könnte, usw. Wenn alle Kredite zurückgezahlt werden könnten, ist der Zustand sicher