VO 182.711 16. Januar 2015

Prüfung Betriebssysteme

KNr. MNr. Zuname, Vorname

Ges.)(100)	1.)(30)	2.)(20)	3.)(50)	Zusatzblätter:

Bitte verwenden Sie nur dokumentenechtes Schreibmaterial!

1 Synchronisation mit Semaphoren (30)

Der Betrieb eines kleinen Skigebiets soll in einer Simulation mit drei Arten von Prozessen, *Gondel*, *Skifahrer* und *Snowboardfahrer*, simuliert werden.

- *Gondel*: Diesen Prozess gibt es genau einmal im System. Er simuliert die Gondelbahn mit einer Gondel, die immer wieder Skifahrer und Snowboarder von der Talstation zur Bergstation befördert und dann leer zur Talstation zurückkehrt.
  Der Prozess simuliert das Öffnen und Schließen der Gondeltür durch Aufrufe der Funktionen *tuer_oeffnen()* bzw. *tuer_schliessen()*. Die Berg- bzw. Talfahrt der Gondel erfolgen durch Aufrufe der Funktionen *bergfahrt()* bzw. *talfahrt()*.
- *Skifahrer*: Von diesem Prozess kann es beliebig viele Instanzen im System geben. Er simuliert eine Folge von Berg- und Talfahrten eines Skifahrers.
  Die Funktionen *einsteigen()*, *aussteigen()* und *abfahren()* simulieren das Einsteigen in bzw. Aussteigen aus der Gondel sowie das Abfahren des Skifahrers auf der Piste.
- *Snowboardfahrer*: wie *Skifahrer*, allerdings mit Einschränkungen bezüglich der Gondelfahrt (siehe unten).

Ergänzen Sie die auf den Folgeseiten gegebenen Prozesstemplates mit *Semaphoroperationen*, um die Prozesse zu synchronisieren. Dabei sind folgende Punkte zu beachten:

- Die Gondel kann N Fahrgäste aufnehmen. Die Gondel fährt von der Talstation ab, wenn sie voll ist und beginnt ihre Talfahrt erst dann, wenn alle Leute am Gipfel ausgestiegen sind.
- Die Gondel verfügt über ausreichend Halterungen für Skier, aber nur über K Halterungen für Snowboards. Daher dürfen bei jeder Bergfahrt maximal K Snowboardfahrer mit der Gondel fahren. Der Rest sind Skifahrer. Der Betrieb der Gondel darf nicht durch das Warten auf Snowboardfahrer verzögert werden.
- Beim Ein- und Aussteigen kann die Türe der Gondel von maximal zwei Personen gleichzeitig passiert werden.

`Initialisierungen`

```
/** Code Gondel **/

for(;;) {
```



```
  tuer_oeffnen();
```



```
  tuer_schliessen();
```



```
  bergfahrt();
```



```
  tuer_oeffnen();
```



```
  tuer_schliessen();
```



```
  talfahrt();
```



```
}
```

```
/** Code Skifahrer **/

do {
```



```
  einsteigen();
```



```
  aussteigen();
```



```
  abfahren();
```



```
} while (not_tired);
```

```
/** Code Snowboardfahrer **/

do {
```



```
  einsteigen();
```



```
  aussteigen();
```



```
  abfahren();
```



```
} while (not_tired);
```

2 Deadlock (20)

Gegeben sind zwei Prozesse, P_1 und P_2, die die beiden Ressourcen R_1 und R_2 unter Mutual Exclusion verwenden.

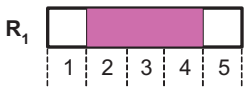
Der Fortschritt von P_1 und P_2 bei der (quasi)parallelen Abarbeitung kann im Prozessfortschrittsdiagramm als Kantenzug zwischen den Punkten *start* und *end* eingetragen werden. Die Achsenbeschriftung entspricht dabei der Zeilennummer des gerade auszuführenden Befehls.

Unterhalb bzw. links der Diagrammachsen sind Balken abgedruckt, in denen die Reservierugen von Ressourcen für P_1 bzw. P_2 eingetragen werden.

Teilaufgabe A

1. Tragen Sie die Reservierungen von Ressourcen für P_1 bzw. P_2 in die entsprechenden Balken ein. Dabei gilt eine Ressource ab Start der Anweisung `get()` als belegt und nach Beendigung der Anweisung `free()` als wieder freigegeben:

```
2: get(R1)
3: ...
4: free(R1)
```



2. Umranden und schraffieren Sie in der Grafik jene Bereiche, durch die der Kantenzug einer (quasi)parallelen Abarbeitung wegen Ressourcenkonflikten nicht möglich ist.

3. Kennzeichnen Sie auf unterschiedliche Weise die Bereiche, die von einem Kantenzug bei einer deadlockfreien Abarbeitung von P_1 und P_2 nicht passiert werden dürfen.

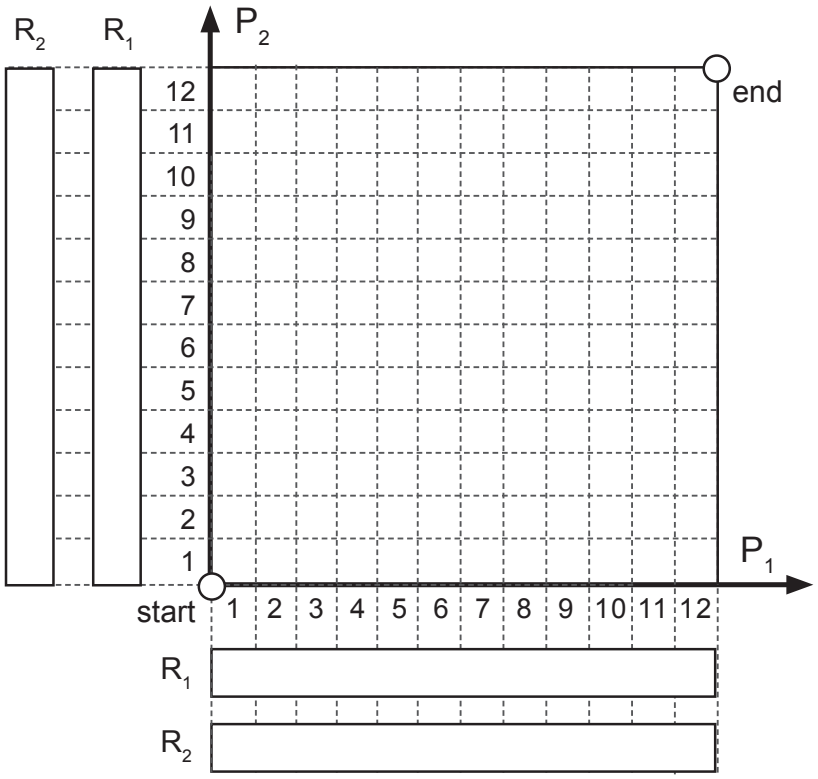
4. Zeichnen Sie einen Kantenzug für eine gültige, deadlockfreie Abarbeitung von P_1 und P_2 in der Grafik ein.

Program P_1:

```
 1: read(buf);
 2: clear(buf2);
 3: get(R1);
 4: copy(R1, buf);
 5: get(R2);
 6: use(R1, R2);
 7: copy(buf2, R2);
 8: free(R2);
 9: print(buf2);
10: clear(buf);
11: free(R1);
12: return;
```

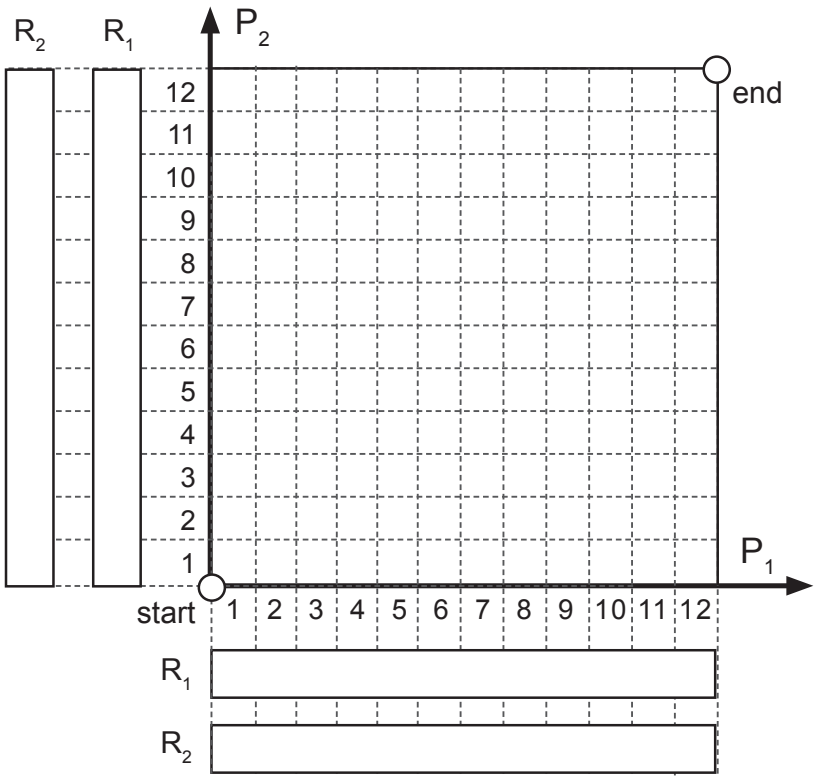
Program P_2:

```
 1: read(buf3);
 2: get(R2);
 3: copy(R2, buf3);
 4: clear(buf4);
 5: get(R1);
 6: copy(buf4, R1);
 7: use(R1, R2);
 8: free(R2);
 9: print(buf4);
10: free(R1);
11: clear(buf3);
12: return;
```



Teilaufgabe B (Deadlock Prevention)

Nehmen Sie nun ein System an, das *Hold and Wait* unterbindet. Tragen Sie die modifizierten Lösungen für die Punkte 1 bis 4 aus Teilaufgabe A im folgenden Diagramm ein.



3 Fragen zu Betriebssystemen (50)

Erklären Sie die Aktionen, die vom Betriebssystem bei einem *Process Switch* durchzuführen sind. Welche Arten von Ereignissen können zu einem Process Switch führen? (3)

Worin liegt der grundlegende Unterschied zwischen *Prozessen* und *Threads*? Welcher Vorteil ergibt sich aus der Einführung von Threads für den Benutzer und worauf muss der Benutzer achten? (4)

Beschreiben Sie die folgenden Strategien für das CPU Scheduling und vergleichen Sie deren Eigenschaften: *Round Robin*, *Virtual Round Robin*, *Highest Response Ratio Next* und *Feedback Scheduling*. (6)

Wir betrachten ein System mit *Virtual Memory Management*. Diskutieren Sie, welche der folgenden Situationen beim Referenzieren einer virtuellen Adresse auftreten bzw. nicht auftreten kann: (a) TLB Miss ohne Page Fault, (b) TLB Miss mit Page Fault, (c) TLB Hit ohne Page Fault, (d) TLB Hit mit Page Fault. (4)

Mit welcher logischen Struktur des I/O Systems versucht man bei der Realisierung von I/O Funktionen sowohl ein einheitliches Programmierinterface, als auch eine möglichst gerätespezifische Ansteuerung zu erreichen? (4)

Was versteht man unter *Synchronous* bzw. *Asynchonous I/O*? Beschreiben Sie die beiden Arten, I/O-Operationen durchzuführen. (3)

Erklären Sie die Begriffe *Access Control List* und *Capability List*. Wozu und wie werden diese verwendet? (4)

Wie berechnet sich die mittlere Zugriffszeit beim Lesen von Daten von einer mechanischen Festplatte? Geben Sie die charakteristischen Zeitparameter einer Festplatte an. Durch welche Strategien kann das Betriebssystem dazu beitragen, die mittleren Zugriffszeiten auf die Festplatte zu reduzieren? (5)

Beschreiben Sie das typische Layout einer Disk bzw. eines Filesystems. Welche Rolle spielen die einzelnen Teile beim Hochfahren des Betriebssystems? (5)

Wie ist ein *i-node* aufgebaut? Welche Informationen enthält er? (4)

Nennen Sie die drei Kategorien von *Security Threats* und beschreiben Sie diese. Geben Sie für jede Kategorie an, welches grundlegende Security-Ziel dadurch bedroht wird. (4)

Nennen Sie Design Prinzipien für die Konstruktion von sicheren Systemen. Geben Sie für jede Regel ein Beispiel an. (4)