Übungen zu Deadlocks

1. Deadlocks (Theorie)

Es gebe fünf Prozesse P_1, P_2, P_3, P_4 und P_5 sowie sechs Ressourcen R_1, R_2, R_3, R_4, R_5, R_6. Es gelte dabei:

- P_1 hat R_2 belegt und fordert R_5 an.
- P_2 hat R_4 belegt und fordert R_2 und R_3 an.
- P_3 hat R_3 belegt und fordert R_1 an.
- P_4 hat R_1 belegt und fordert R_6 an.
- P_5 hat R_5 belegt und fordert R_4 an.

a) Zeichnen Sie den Ressourcen-Zuordnungsgraph für dieses Szenario und leiten Sie daraus ab, ob sich die fünf Prozesse im Deadlock-Zustand befinden. Begründen Sie Ihre Antwort.

b) Überprüfen Sie Ihr Ergebnis, indem Sie den Deadlock-Erkennungs-Algorithmus (mit den Belegungs- und Anforderungsmatrizen) durchführen.

2. Deadlocks (Praxis)

Betrachten Sie das auf Seite 2 abgedruckte Programm, das Sie auch als `deadlock.c` auf der Webseite finden.

a) Ohne aktiviertes Cheating läuft dieses Programm in einen Deadlock. Woran liegt das und wie könnten Sie das Problem lösen?

b) Testen Sie das Programm in der VM – einmal mit und einmal ohne aktiviertes Cheating.

c) Schalten Sie das Cheating wieder aus und korrigieren Sie die Synchronisation (wie in Teil a)).

d) Welchen Nachteil hat die korrekte Implementierung?

```
// deadlock.c
#include <stdio.h>
#include <pthread.h>

pthread_t       thr1,  thr2;
pthread_mutex_t res_A, res_B;

void *thread1 (void *args) {
  pthread_mutex_lock (&res_A);    printf ("thread1: locked A\n");
  pthread_mutex_lock (&res_B);    printf ("thread1: locked B\n");

  printf ("thread1: critical!\n"); sleep (1);

  pthread_mutex_unlock (&res_A);  printf ("thread1: unlocked A\n");
  pthread_mutex_unlock (&res_B);  printf ("thread1: unlocked B\n");
}

void *thread2 (void *args) {
  pthread_mutex_lock (&res_B);    printf ("thread2: locked B\n");
  pthread_mutex_lock (&res_A);    printf ("thread2: locked A\n");

  printf ("thread2: critical!\n"); sleep (1);

  pthread_mutex_unlock (&res_A);  printf ("thread2: unlocked A\n");
  pthread_mutex_unlock (&res_B);  printf ("thread2: unlocked B\n");
}

// Folgende Zeile auskommentieren, um Cheating zu aktivieren
// #define CHEATING

int main () {
  pthread_mutex_init(&res_A, NULL);
  pthread_mutex_init(&res_B, NULL);

  // Threads erzeugen
  pthread_create (&thr1, NULL, thread1, NULL);
  pthread_create (&thr2, NULL, thread2, NULL);

  #ifdef CHEATING
    // Cheating
    sleep (5);
    printf ("main: Nach 5 Sekunden: Cheat (unlocking A)...\n");
    pthread_mutex_unlock (&res_A);    // illegal!
  #endif

  // Threads einsammeln
  pthread_join (thr1, NULL);
  pthread_join (thr2, NULL);

  printf ("Fertig\n");
}
```