Problem Set 12

CS 411

Due at the beginning of class on the first class day of the following week.
Sections 8.5–8

1. In a real computer system, neither the resources available nor the demands of processes for resources are consistent over long periods (months). Resources break or are replaced, new processes come and go, and new resources are bought and added to the system. If deadlock is controlled by the banker's algorithm, which of the following changes can be made safely (without introducing the possibility of deadlock), and under what circumstances?

   (a) Increase ***Available*** (new resources added).

   (b) Decrease ***Available*** (resources permanently removed from the system).

   (c) Increase ***Max*** for one process (the process needs or want more resources than allowed).

   (d) Decrease ***Max*** for one process (the process decides it does not need that many resources).

   (e) Increase the number of processes.

   (f) Decrease the number of processes.

2. Consider the following snapshot of a system:

	Allocation				***Max***				***Available***			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	2	0	0	1	4	2	1	2	3	3	2	1
P_1	3	1	2	1	5	2	5	2				
P_2	2	1	0	3	2	3	1	6				
P_3	1	3	1	2	1	4	2	4				
P_4	1	4	3	2	3	6	6	5				

   Answer the following questions using the banker's algorithm. Show work.

   (a) Illustrate that the system is in a safe state by demonstrating an order in which the processes can complete.

   (b) If a request from process P_1 arrives for $(1, 1, 0, 0)$, can the request be granted immediately?

   (c) If a request from process P_4 arrives for $(0, 0, 2, 0)$, can the request be granted immediately?

3. How could two threads become deadlocked executing the following `transaction` function? Fix this function to prevent deadlocks from occurring.

```
void transaction(Account from, Account to, double amount)
{
  mutex lock1, lock2;
  lock1 = get lock(from);
  lock2 = get lock(to);

  acquire(lock1);
    acquire(lock2);

      withdraw(from, amount);
      deposit(to, amount);

    release(lock2);
  release(lock1);
}
```