COM S 352
Assignment 5
Due: October 13, 2017

7.15 Compare the circular-wait scheme with the various deadlock-avoidance schemes (like the banker's algorithm) with respect to the following issues:
a. Runtime overheads
b. System throughput

7.16 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*** (resource permanently removed from system).
c. Increase ***Max*** for one process (the process needs or wants more resources than allowed).
d. Decrease ***Max*** for one process (the process decides it does not need that many resources).

7.18 Consider a system consisting of *m* resources of the same type being shared by *n* processes.Aprocess can request or release only one resource at a time. Show that the system is deadlock free if the following two conditions hold:
a. The maximum need of each process is between one resource and *m* resources.
b. The sum of all maximum needs is less than *m* + *n*.

7.22 Consider the following snapshot of a system:

	Allocation				***Max***			
	A	*B*	*C*	*D*	*A*	*B*	*C*	*D*
*P*0	3	0	1	4	5	1	1	7
*P*1	2	2	1	0	3	2	1	1
*P*2	3	1	2	1	3	3	2	1
*P*3	0	5	1	0	4	6	1	2
*P*4	4	2	1	2	6	3	2	5

Using the banker's algorithm, determine whether or not each of the following states is unsafe. If the state is safe, illustrate the order in which the processes may complete. Otherwise, illustrate why the state is unsafe.
a. ***Available*** = (0, 3, 0, 1)
b. ***Available*** = (1, 0, 0, 2)

7.24 What is the optimistic assumption made in the deadlock-detection algorithm? How can this assumption be violated?

6. The following restrictions apply:
1) One thread will print "hello ", one thread will print "world", and the main function will print the trailing "\n",
2) Ensure thread synchronization using pthread mutex

```
/* Include Files */
#include <stdio.h>
/* External References */
extern void world( void );
extern void hello( void );
void main( int argc, char *argv[] ) {
world();
hello();
printf( "\n" );
}
/* world - print the "world" part. */
void world( void ) {
printf( "world" );
}
/* hello - print the "hello" part. */
void hello( void ) {
printf( "hello " );
}
```