

**Problem 1: Definitions**

Define each of the following terms:

- Deadlock:
- Livelock:
- Priority inversion:
- Working set:

**Problem 2: Deadlocks**

a) Consider the following three threads executing different paths of execution. Determine if and where deadlock can occur for each thread. For example, “Thread 1 can deadlock on Line 2” would mean that Thread 1 can deadlock and wait forever to acquire mutex\_e.

Thread 1	Thread 2	Thread 3
lock(mutex_d)	lock(mutex_e)	lock(mutex_f)
lock(mutex_e)	lock(mutex_f)	lock(mutex_d)
e=d*2+e	f=f+e	f=f*d
unlock(mutex_e)	unlock(mutex_f)	unlock(mutex_d)
unlock(mutex_d)	unlock(mutex_e)	unlock(mutex_f)

Complete the following, or write “deadlock cannot occur”

Thread 1 can deadlock on line \_\_\_\_\_

Thread 2 can deadlock on line \_\_\_\_\_

Thread 3 can deadlock on line \_\_\_\_\_

b) Repeat part a) for the following two threads.

Thread 1	Thread 2
lock(mutex_h)	lock(mutex_g)
lock(mutex_i)	lock(mutex_i)
lock(mutex_j)	i = g + g
j = i + h	unlock(mutex_i)
unlock(mutex_j)	unlock(mutex_g)
unlock(mutex_h)	
lock(mutex_g)	
g=g+i	
unlock(mutex_g)	
unlock(mutex_i)	

Complete the following, or write “deadlock cannot occur”

Thread 1 can deadlock on line \_\_\_\_\_

Thread 2 can deadlock on line \_\_\_\_\_

**c)** Given the following processes and their resource allocations, use the banker’s algorithm to determine if a deadlock is inevitable; show your calculations. The system has 10 units of X and 15 units of Y.

Process	has X	may need X	has Y	may need Y
1	3	10	2	4
2	0	6	6	7
3	5	5	2	6
4	1	2	5	5

**d)** Give an example of a resource allocation graph that includes a cycle, but does not include a deadlock.

**Problem 3: Address Translation**

a) In a memory system that uses paging, what are the advantages and disadvantages of using larger pages over smaller pages?

b) Assume you have a system with 32-bit addresses, and suppose that you want to have 4 KB pages.

How many bits do you need to represent offsets within a page?

How many bits would you need to represent a virtual page number?

c) In the same system (with 4 KB pages and 32-bit addresses), suppose that you have a TLB of size 3 entries containing the following mappings:

Virtual page number	Physical page number	Valid bit
0x31415	0x01337	1
0xCAFE5	0xF000D	1
0xDECAF	0xC0FEE	0

For each of the following virtual addresses accessed by the program, determine whether there would be a TLB hit or miss, and for the hits, write the physical address that the virtual address translate to.

Virtual address	Hit or miss?	Physical address
0x31415926		
0xCAFE1335		
0xDECAF123		
0x31415555		

**Problem 4: Short Answers**

**a)** A hardware test-and-set instruction provides a simple way to achieve mutual exclusion using only user-level code. Why might a kernel-level solution still be preferable?

**b)** Without hardware support for test-and-set, could you still implement critical sections? How would you do it?

**c)** What is the worst possible CPU scheduling algorithm, i.e. the one that yields the worst possible average response time? Assume that the algorithm must run a thread if there is one available, that a context switch has zero overhead, and that the length of each job is known when it is submitted.