1) A system is composed of four processes, \( p_1 \) through \( p_4 \), and three types of consumable resources, \( R_1 \) through \( R_3 \). There is one unit each of \( R_1 \) and \( R_3 \) available.

- \( p_1 \) requests a unit of \( R_1 \) and a unit of \( R_2 \).
- \( p_2 \) produces a unit of \( R_1 \) and a unit of \( R_2 \) and requests one unit of \( R_3 \).
- \( p_3 \) requests a unit of \( R_1 \) and a unit of \( R_2 \).
- \( p_4 \) produces a unit of \( R_2 \) and requests one unit of \( R_3 \).

Show the resource allocation graph to represent the system state. Which, if any, of the processes are deadlocked in this state? Can the deadlock be avoided by some sequence of execution of the processes?

2) Consider a multi-level feedback queue in a single-CPU system. The first level (queue 0) is given a quantum of 8 ms, the second one a quantum of 16 ms, the third is scheduled FCFS. Assume jobs arrive all at time zero with the following job times (in ms): 4, 7, 12, 20, 25 and 30. Show the Gantt chart for this system and compute the average waiting and turnaround time.

3) Compare the LOOK and C-LOOK disk scheduling algorithms. Which might one prefer over the other?

4) Consider a system with 64 MB of physical memory, 32-bit physical addresses, 32-bit virtual addresses, and 4 KB physical page frames.

   a) Using a single-level paging scheme, what is the maximum number of page table entries for this system?
   b) Using a two-level paging scheme with a 1024-entry outer-page table, what would be the page offset of the page of the page table accessed for the virtual address 001101000010000110001010011101?
   c) Suppose a TLB is used with the two-level paging scheme described in part b, and the TLB has a 90% hit rate. If the TLB access time is 10 ns and memory access time is 100 ns, what is the effective memory access time of the system?
   d) If an inverted page table is used to translate virtual addresses to physical addresses, how large would it need to be?
e) What disadvantage is there, if any, in using an inverted page table versus per process page tables?

5) The local laundromat has just entered the computer age. As each customer enters, he or she puts coins into slots at one of two stations and types in the number of washing machines he/she will need. The stations are connected to a central computer that automatically assigns available machines and outputs tokens that identify the machines to be used. The customer puts laundry into the machines and inserts each token into the machine indicated on the token. When a machine finishes its cycle, it informs the computer that it is available again. The computer maintains a boolean array `available[NMACHINES]` to represent if corresponding machines are available (NMACHINES is a constant indicating how many machines there are in the laundromat), and a semaphore `nfree` that indicates how many machines are available. The code to allocate and release machines is as follows. The `available` array is initialized to all true, and `nfree` is initialized to NMACHINES.

```c
int allocate() /* Returns index of available machine.*/
{
    sem_wait(nfree); /* Wait until a machine is available */
    for (int i=0; i < NMACHINES; i++)
        if (available[i] != TRUE)
        {
            available[i] = FALSE;
            return i;
        }
}

void release(int machine) /* Releases machine */
{
    available[machine] = TRUE;
    sem_signal(nfree);
}
```

(a) It seems that if two people make requests at the two stations at the same time, they will occasionally be assigned the same machine. This has resulted in several brawls in the laundromat, and you have been called in by the owner to fix the problem. Assume that one thread handles each customer station. Explain how the same washing machine can be assigned to two different customers.

(b) Modify the code to eliminate the problem.
6) Answer the following questions true or false:

a. An operating system can be viewed as a "resource allocator" to control various I/O devices and user programs.

b. The following instructions must be protected to ensure that a computer system operates correctly: change to monitor mode, read from monitor memory, write into monitor memory, and turn off timer interrupts.

c. I/O instructions and turning interrupts on are not generally considered to be privileged instructions.

d. Deadlock can be prevented in the dining philosophers problem by simply reducing the number of philosophers that are allowed to eat at the same time by one.

e. On a uniprocessor system, the critical section problem can be solved simply by disabling interrupts while a shared variable is being modified.

f. A thread is generally more lightweight than a process because threads have their own virtual address spaces while processes may have shared address spaces.

g. A priority scheduler can be used to implement any other kind of scheduler, given an appropriate choice of priorities and the ability to change those priorities at any time.

h. When using a text editor in UNIX, the characters that are typed are processed by the text editor application without any operating system activity other than processor scheduling.

i. While LRU page replacement is more complicated than FIFO page replacement, LRU is generally used because it performs at least as well as FIFO for all possible memory reference strings.

j. A deadlock cannot arise for a set of processes unless there is a circular wait condition.