Instructions: There are **five** questions. Each question may have several sub-questions. Please read the questions carefully before answering the questions.

1. (20pts) For each process, the operating system uses the Process Control Block (PCB) to store the process information. Threads of the one process shares the same memory space. Please answer the following questions:
   a. Please define Thread Control Block and Process Control Block and compare the difference in terms of the size of the TCB and PCB.
   b. In some applications such web servers, multithreading may achieve better performance. However, thread creation incurs overhead. How can we eliminate the thread creation overhead to improve performance?
   c. Alice is asked to design a new banking service which accepts requests to query the account balance, fund transfer, etc. The service requires higher security level than other systems. Should Alice adapt multi-thread or multi-process model to design the new service? Please explain your answer.

2. (20pts) Bob designed a new resource synchronization protocol for a priority-driven scheduling algorithm. In his protocol, each process is assigned a priority level, called *process priority level*, and several processes may have the same process priority level. In addition, each resource also has a priority level called *resource priority level*. Before the system starts, the resource priority level of each resource is set as the greatest priority level among that for the processes may request the same resource. Bob's protocol is the following:
   a. A process can lock a resource if and only if its process priority level is ***greater*** than the resource priority of other resources already locked by other processes,
   b. When a process is blocked by other lower priority processes, the process priority level of the lower priority process is set as its process priority level.

   Bob claims that his protocol can avoid deadlock. Is Bob correct? If yes, please prove that there is no deadlock. If no, please use a counter example to prove that Bob's protocol cannot avoid deadlock.

3. (20pts) Catherine is given a Bounded-buffer Producer-Consumer Problem to solve. The assignment is the following. A buffer of size *n* is shared by several processes. We are given sequential code *add_item*(): adds an item to the buffer and *remove_item*() removes an item from the buffer. We want to implement functions Append(item) and Remove(item) such that the following hold:
   a. *Mutually exclusive access to buffer*: At any time at most one process should be executing *add_item()* or *remove_item()*.
   b. *No buffer overflow*: A process executes *add_item*() only when the buffer is not full (i.e., the process waits if the buffer is full).
   c. *No buffer underflow*: A process executes *remove_item*() only when the buffer is not empty.
   d. *No busy waiting*.
   e. *No producer starvation*: A process does not wait forever at Append() provided the buffer repeatedly becomes non-full.

f. *No producer starvation*: A process does not wait forever at Remove() provided the buffer repeatedly becomes non-empty.

Catherine uses the binary and counting semaphore to implement the *Append*() and *Remove*(). Before Catherine submits her homework, please check if her program meets the requirements. If yes, please prove the above program solve the producer and consumer program. If no, please help her to debug the program.

```
Semaphore nItem initially 0;  // number of items in the buffer
Semaphore mutex initially 1;  // protects add_item and remove_item

Append( item ) {
  P( mutex );
  add_item;
  V( mutex );
  V( nItem );
}

Remove( item ) {
  P( nItem );
  P( mutex );
  remove_item;
  V( mutex );
}
```

4. (20 pts) Daniel proposed a new scheduling algorithm and claims that the algorithm favors *new* processes and is fair to *old* processes. In Daniel's scheduling algorithm, there are two queues: *new process queue* and *old process queue*. When a new process arrives, the process is inserted into the new process queue. When a new process has been in the new process queue for 10 milliseconds, the process is removed from the new process queue and inserted into the old process queue. When it is time to schedule, the scheduler selects one process from the head of new process queue when new process queue is not empty. Otherwise, it selects one process from the head of the old process queue. Each process will be given a 10-milliseconds time slot. (NOTE: You will receive only one point if you can only correctly answer the questions with YES or NO.)

   a. (5 pts.) Is this algorithm starvation free? If yes, please explain your answer. If not, please give an example showing that a process may starve.

   b. (5 pts.) The mechanism of moving new processes to the old process queue can be implemented by setting timers for every process. However, it incurs heavy overhead to the system. Can this mechanism be implemented without timer?

c. (5 pts.) Discuss whether the algorithm favors new processes?

d. (5 pts.) Discuss whether the algorithm is *fair* to all processes? By fair, we mean that every process has a wait time approximately equal to the average wait time, assuming all processes have close to the same execution time. Please discuss for the following two cases.

   i. (2 pts) Assume that the execution times of the processes are much less than 10 milliseconds.

   ii. (3 pts) Assume that the execution times of the processes are much longer than 10 milliseconds.

5. (20 pts) In distributed systems, there are several ways of exchanging packets between computing nodes over the network. The three most common packet exchange schemes are *circuit switching*, *message switching*, and *packet switching.*

   a. (3 pts) Please define circuit switching scheme.

   b. (3 pts) Please define message switching scheme.

   c. (4 pts) Please define packet switching scheme.

   d. (5 pts) It is often argued that the circuit switch scheme may cause low network bandwidth utilization but the packet switch scheme does not. Please discuss it.

   e. (5 pts) Circuit switch scheme offers more reliable communication. Can the circuit switch be implemented using packet switching scheme?