1. (40%, 5% each) Answer each of the following questions concisely but precisely.
   
   - Why do we still disable interrupts in addition to using the lock with TAS in the following implementation of wait() for multiprocessors?

   ```c
   void wait(semaphore s)
   {
       disable interrupts;
       while (TAS(s->lock, 1) == 1);
       if (s->count > 0) { s->count--; s->lock = 0; enable interrupts; return; }
       add(s->q, current_thread); s->lock = 0; enable interrupts; sleep();
   }
   ``

   - Can Round Robin ever be the worst possible CPU scheduling algorithm in terms of turn around time? If so, under what circumstances? If not, explain why not.

   - What are the tradeoffs in choosing the page size in a paged main memory management system?

   - In a paging system, when malloc(16385) invoked by user process P successfully returns, how many physical pages have been allocated to process P, assuming a page size of 4KB? Are they consecutive in the physical memory?

   - In a paging system, what is the motivation for using an inverted page table?

Write in Exam Book Only
• What is the challenge/downside in using an inverted page table?

• What is copy-on-write in virtual memory?

• Why do many file systems include facilities for *links*? What must a file system without links do to accomplish the same function?
2. (Process Synchronization - 20%)

The following "solution" to the mutual exclusion for two processes was published in 1966 in the Communications of the ACM.

Process 1 invokes bogusMutex(0) and process 2 invokes bogusMutex(1). Only individual load and store instructions can be assumed to be atomic.

```java
boolean blocked[2] = {FALSE, FALSE};
int turn = 0;

1: void bogusMutex(int pid) {
2:   while (1) {
3:     blocked[pid] = TRUE;
4:   while (turn != pid) {
5:     while (blocked[1-pid]);
6:   }
7:   turn = pid;
8: }

9:   [critical section]

10:   blocked[pid] = FALSE;
11: }
12: }
```

Come up with a sequence of executions of the two processes' statements using the line numbers above (i.e., find a counter example) that demonstrates that this solution is incorrect. Show the exact sequence of events that leads to incorrect behavior.
3. (CPU Scheduling - 16%) Consider a preemptive priority scheduling algorithm based on dynamically changing priorities. Larger priority numbers imply higher priority. When a process is waiting for the CPU (in the ready queue, but not running), its priority changes at a rate $\alpha$; when it is running, its priority changes at a rate $\beta$. All processes are given a priority of 0 when they enter the ready queue. The parameters $\alpha$ and $\beta$ can be set to give many different scheduling algorithms.

(a) (8%) What is the algorithm that results from $\beta > \alpha > 0$?

(b) (8%) What is the algorithm that results from $\alpha < \beta < 0$?
4. (Redundant Arrays of Inexpensive Disks - 24%) 
   
   (a) (8%) Name two possible advantages of RAID over single disks. 
   
   (b) Consider a RAID Level 5 that performs block-level striping over 8 disks for data and uses 1 (additional) disk for parity blocks. 
      
      i. (8%) How many block reads and writes happen in writing a large file with 8 blocks (assuming they are perfectly striped over the 8 disks for data). Explain which reads/writes can happen in parallel. 
      
      ii. (8%) How many block reads and writes happen in writing a small file with 1 single block. Explain which reads/writes can happen in parallel.