1. (30%, 3% each) Answer each of the following questions concisely but precisely.

- Why is the busy-waiting in the following implementation of semaphore wait() on multiprocessors not considered a problem?

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

- Why do we still disable interrupts (to prevent context switch from happening) 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 average turn around time? If so, under what circumstances? If not, explain why not.

- In a generic storage allocation problem, what are the fundamental causes for external fragmentation?
• List at least two benefits of using a two-level page table compared to a one-level page table?

• What is an inverted page table?

• What is the challenge/downside in using an inverted page table?

• In a paging system, when `malloc(8193)` invoked by user process $P$ successfully returns, how many physical pages have been allocated to process $P$, assuming a page size of 4096 bytes? Are they consecutive in the physical memory?

• Why are TLBs often implemented as fully set-associative, in contrast to an L1 cache that is generally two-way set-associative or direct-mapped?

• What is a page fault?
2. (True-or-False - 21%, 3% each) For each of the statements below, indicate in one sentence whether or not the statement is true or false, and why.

- The use of a log stored in fast, non-volatile RAM increases the reliability but not the performance of a file system.

- A multi-level indexed file descriptor permits faster random access than a file descriptor with a single level of index.

- (Bonus 2%) In the Unix File System, user processes can directly write directories using the write() system call just like ordinary files.

- In the Unix File System, the name of a (non-directory) file is stored in its parent directory’s inode.

- In File Systems, the buffer cache is implemented purely in software, unlike demand paging, which is implemented jointly in software and with hardware support (i.e. MMU).

- It is slower to write 1 block in a RAID Level 5 organization with 5 disks than in a RAID Level 1 organization with 2 disks (i.e., mirroring only).

- In disk scheduling, the Shortest-Seek-Time-First (SSTF) scheduling algorithm tends to favor middle cylinders over the innermost/outmost cylinders.
3. (Deadlock - 20 pts) On planet Venus, life is really boring that all resources are of the same type. Consider a system where $m$ resources are shared by $n$ processes. Resources can be acquired and released only one at a time, and can be used by only one process at a time. For example, if a process needs to acquire two resources, it cannot acquire two atomically.

Show that deadlock cannot occur, as long as the maximal resource need of each process is between 1 and $m$, and the sum of all maximal needs is less than $m + n$. 
4. (Directories and Links - 20%)

(a) Why do many file systems include facilities for links? What must a file system without links do to accomplish the same function?

(b) Given an absolute pathname, "/homes/user1/exam2.txt", what are the disk reads performed by UFS in order to read the first data block of the file? Assume no data or metadata blocks are cached, and that each directory file contains only 1 data block.

(c) If "/hardlink" is a hard link that points to "/home/user1/exam2.txt", using "/hardlink", what are the disk reads performed by UFS in order to read the first data block of the file? Assume no data or metadata blocks are cached, and that each directory file contains only 1 data block.

(d) If "/softlink" is a soft link that points to "/home/user1/exam2.txt", using "/softlink", what are the disk reads performed by UFS in order to read the first data block of the file? Assume (1) the inode for "/" is cached in memory; (2) no other data or inode blocks are cached, but once loaded, they are cached; (3) each directory file has only one data block, which contains information for all the files in it; (4) the file name pointed to by a softlink is stored in its inode.