1. (28 pts, 4 pts each) Answer each of the following questions concisely but precisely.

   - What is a system call?

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

```
void wait(semaphore s)
{
   disable interrupts;
   while (ldl(s->lock) != 0 || !stc(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();
}
```


   - Can FIFO ever be the worst possible non-preemptive 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 fundamentally causes external fragmentation?

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

- In a paging system, when malloc(16384) invoked by user process P successfully returns, how many physical pages have been allocated to process P, assuming a page size of 4KB? If the answer is more than 1, 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?

2. (Synchronization - 12 pts) In pseudo-code below, outline how to use the `P()` and `V()` primitives to solve the Producer/Consumer problem with a pool of empty buffers and a pool of full buffers. Assume there are N empty buffers initially, and no full buffers.

   Be careful to indicate which semaphores you are using where; note which kind each semaphore is (from your answer above), and the value it is initialized with.

   No partial credit for solutions that either have deadlock or races.

```
Producer:
   // Put producer
   // pseudocode here.

while(1)
{


   get empty buffer
       from pool of empties



   produce data in buffer



   put full buffer
      in pool of fulls


}
```

```
Consumer:
   // Put consumer
   // pseudocode here.

while(1)
{


   get full from pool of fulls



   consume data in buffer



   put empty buffer
      in pool of empties


}
```

Write in Exam Book Only

3. (Deadlock - 20 pts) On planet Mars, 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.

Write in Exam Book Only

4. (Paging and Segmentation - 20 pts)

   A certain 32-bit architecture uses segmentation with paging to translate a 32-bit *virtual address* into a 32-bit *physical address*. The processor uses the first 4 bits of the virtual address to select the segment and thus the one-level page table for that segment, the next 16 bits of the virtual address to provide the offset into the one-level page table, and finally the last 12 bits as the offset within each 4KB page. Assume there is a root pointer register, that holds the base address for the segment table.

   (a) (10 pts) Diagram the data structures involved in translating a virtual address down to a final physical address. Label each table, and make clear how each step of translation takes place.

   (b) (5 pts) How many memory accesses does this lookup take in the worst case? What are they?

   (c) (5 pts) How much memory is taken up by the segment table and the page tables for a full-sized process (i.e. the entire 4GB is used)? 5

5. (UFS - 20%)

(a) (4%) What is an inode? List at least 3 pieces of information typically stored in it.

(b) (4%) What is a superblock? List at least 3 pieces of information typically stored in it?

(c) Consider a UNIX File System implementation with a logical block size of 512 bytes, and a traditional 128-byte inode containing, among other things, 10 direct block pointers, single-, double- and triple-indirect pointers. Assume 32-bit pointers.

i. (6%) What is the maximum number of blocks of metadata overhead that a large file can have? (no partial credit for incorrect answer)

ii. (6%) A process seeks to and reads the 70,659th byte of a large file. (This is byte number 0x00011403 in hexadecimal. A file starts with the first byte, which has a bytenumber 0x00000001 in hexadecimal.) Assuming no metadata is cached, how many disk reads total does this access take once? Note that the process must already have an inode number for the file. Diagram the metadata structures accessed in order to read this file block. Include index numbers in any of the tables.

Write in Exam Book Only