The exam is out of 250. You get 75 points as a holiday gift. Answer 175 points worth of questions; if you answer more than that, I will throw away/scale down your worst answers. If you answer a question you must answer all required parts. Please try to be concise; this is an open book exam – you will get no credit for repeating information when it is not relevant.

**Problem 1:** (20 pts)
Throughout this class, we discussed different subsystems in the operating system. List four of these subsystems. Pick one of them, and briefly discuss the important concepts and design decisions that an OS designer has to make in it. Discuss its relationship to the other subsystems.

**Problem 2:** (15 points) Much of the time in this class was spent on “scheduling policies” (policies that pick a candidate out of a pool of available candidates). Describe three situations where such policies were needed. Discuss the criteria for a good policy in each situation and suggest a good policy.

**Problem 3:** (24 pts)
Provide short explanations for four of the following potentially wrong statements. Only four will count for credit if you answer more.
1. A multithreaded program will run faster than a single threaded one.
2. Page replacement would not be necessary if we use a disk cache
3. Mixing segmentation and paging by paging segments is useful; mixing them by segmenting pages is not
4. If programs had a random reference pattern to memory (no locality), a TLB would be useless.
5. To ensure security in a system, you should use prevention (making sure no one can do anything bad) techniques or detection (discovering someone who is trying to do something bad) techniques, but not both.

**Problem 4:** (32 pts)
Briefly answer four of the following questions. Only four will count for credit if you answer more.
1. Are semaphores (or other synchronization primitives) needed to implement the bakery algorithm? Explain. Show whether or not the bakery algorithm is fair.
2. If a programmer knew how the scheduler in the OS works, is there anything they can do to make their program get a bigger share of the CPU? Explain using Multiple-Level Feedback as your scheduling algorithm. Would the same apply to Round Robin?
3. Memory protection techniques such as the limit register in segmented memory prevent illegal access to data. Similarly, mutexes prevent illegal accesses to shared data by concurrent threads. However, memory protection is mandatory, whereas the protection by mutexes is voluntary. Explain.
4. Construct a safe resource allocation state in which there are three processes, there are two classes of resources with 4 resources available of each type. Also, provide a request that, if honored, would lead to an unsafe state.
5. The X-FS (file system from silicon graphics), uses a block size that is equal to the system page size (4Kbytes for the pentium). Why would this be useful?

**Problem 5:** (30 points) The size of the block on a disk drive is fixed. This makes drives similar to paged memory where the frame is of fixed size. An engineer in your company would like to build a disk organization where the block size is variable (like segments). Outline the challenges involved in building such a system. Discuss the implications on issues such as directory organization, file allocation, free space management, disk scheduling and disk caching.

**Problem 6:** (30 points) A group of students are studying for the CS350 final. The students can study only while eating pizza. Each student executes the following loop: while (1) pick up a piece of pizza; study while eating the pizza. If a student finds that the pizza is gone, the student goes to sleep until another pizza
arrives. The first student to discover that the group is out, phones Nirchi’s to order another pizza before going to sleep. Each pizza has S slices. Write code to synchronize the student threads and the pizza delivery thread. Your solution should avoid deadlock and phone Nirchi’s exactly once each time a pizza is over. No pice of pizza can be consumed by more than one student.

Problem 7: (30 points) Tweedledum and Tweedledee are separate threads executing their respective procedures. The code below is intended to cause them to forever take turns exchanging insults through the shared variable X in strict alternation. The Sleep() and Wakeup() routines operate as follows. Sleep blocks the calling thread, and Wakeup unblocks a thread if it is asleep, otherwise it does nothing.

```c
Tweedledum() {
    while(1) {
        Sleep();
        x=Insult();
        Wakeup(Tweedledee);
    }
}

Tweedledee() {
    while(1) {
        Sleep();
        x=Insult();
        Wakeup(Tweedledum);
    }
}
```

(a) The code shown above has a synchronization flaw. Briefly outline a scenario in which this code would fail and the outcome of the scenario.

(b) Show how to fix the problem replacing Sleep() and Wakeup() with semaphore calls. You may not disable interrupts or use Yield().

(c) Assuming that the threads above are User Level Threads, which of the operations above would require a system call. Explain.

Problem 8: (25 pts)
(a) A student majoring in anthropology and minor in computer science has embarked on a research project to see if African baboons can be taught about deadlocks. He locates a deep canyon and fastens a rope across it, so the baboons can cross hand-over-hand. Several baboons can cross at the same time provided that they are all going in the same direction. If eastward moving and westward moving baboons ever get onto the rope at the same time, a deadlock will result because it is impossible for one baboon to climb over another one while suspended over the canyon. Show that the ingredients for deadlock are present.

(b) Two teams are playing tug of war (each team is pulling on one end of the rope, a team wins when it drags the other team a long enough distance for them to fall in a puddle. At the beginning the puddle is in the middle of the two teams). The two teams are of equal strength. Is this an example of deadlock?

Problem 9: (30 pts)  
Virtual memory can be thought of as a cache for the disk drive.

(a) (6 points) Explain the above statement.

(b) (6 points) Often, there is also a separate disk cache set aside in memory. Is this redundant?

(c) (6 points) List two other caches in a computer system.

(d) (12 points) Replacement is usually an issue in cache design. Do all caches need a replacement policy? For two of the caches discussed above, suggest a suitable replacement policy. Justify your answer.

Good Luck, and have a great Summer!