Hints:

- Write your name and student number on the blue book NOW!
- The exam is open book.
- You have 120 minutes for the exam. Be a smart exam taker: all points are not born equal. So, if you get stuck on one problem, move on to another. Also, don’t waste your time giving irrelevant (or not requested) details or material. If you say too much and what you say is partially wrong, it may hurt you.
- Clearly identify the question that you are answering for each answer.
- Show all your work. Partial credit is possible for an answer, but only if you show the intermediate steps in obtaining the answer.
- Give answers only in the blue book.

Questions:

1. [10pts] Here is the description of a class Communicator that my undergraduate class had to build last semester as part of the Nachos system. (Nachos is an instructional operating system.)

   Implement synchronous send and receive of one word messages between two threads using condition variables. Implement the Communicator class with operations, `void speak(int word)` and `int listen()`. `speak()` atomically waits until `listen()` is called on the same Communicator object, and then transfers the word over to `listen()`. Once the transfer is made, both can return. Similarly, `listen()` waits until `speak()` is called, at which point the transfer is made, and both can return (`listen()` returns the word).

   What is wrong with the following (partial) implementation of Communicator and why? (Many of the students actually had code that look very much like this.)

   ```java
   public class Communicator {
       Lock lock;
       Condition waitForSpeaker
       ConditionwaitForListener;
       Condition waitForBuffer;

       int buffer; // buffer used to transfer word
       boolean bufferUsed = false; // is the buffer in use?

       void speak(int word) {
           lock.acquire();
           if (bufferUsed) // wait for buffer to become free
               waitForBuffer.sleep();
           bufferUsed = true;
           buffer = word;
           waitForSpeaker.wake(); // wake any waiting listener
           waitForListener.sleep(); // wait for a listener
           lock.release();
       }

       int listen() {
           lock.acquire();
           if (!bufferUsed) // wait for a speaker
               waitForSpeaker.sleep();
           lock.release();
           return buffer;
       }
   }
   ```
int word = buffer;
bufferUsed = false;
waitForBuffer.wake(); // wake speaker waiting for buffer
waitForListener.wake(); // wake speaker waiting for listener
lock.release();
return word;
}

2. [10pts] The SPARC instruction set contains an instruction "SWAP reg,mem" which atomically exchanges the values in one of the registers and a memory location. E.g. if register 12 contained the number 4 and memory cell 100 contained the number 2, after executing "SWAP r12,100" register 12 would contain the number 2 and memory cell 100 would contain the number 4. Assume that you are given the following swap() function implemented using the SWAP instruction:

```c
int swap(newval, var)
// set var to newval
// return old value of var (before the swap)
```

Write the pseudo-code which implements a lock using swap(). A spin-lock is acceptable. (Remember, a lock has two methods, acquire() and release().)

3. [20pts] Santa Claus sleeps in his shop at the North Pole and can only be wakened by either (1) all nine reindeers being back from their vacation in the South Pacific, or (2) by some of the elves having difficulties making toys. When an elf has a problem, he should: (a) wake Santa if Santa is sleeping, and (b) get in-line outside Santa’s door. When a reindeer arrives, he should get in-line outside of Santa’s door. The last arriving reindeer should wake Santa up if Santa is sleeping. When Santa first wakes up or when he finishes dealing with an elf’s problem, he looks outside his door. If the last reindeer is back, Santa will go get his sleigh ready and go distribute the toys. Otherwise, if one or more elf is waiting, he let’s one elf into his room to discuss the problem. Finally, if there are no waiting elves and all nine reindeers are not back, Santa goes back to sleep. Write three routines, Reindeer, Elf, and Santa to synchronize the above activities using any combination of locks, semaphores, and condition variables. You are NOT allowed to manipulate interrupts and/or queues of TCBs.

4. [20pts] Consider a multilevel feedback queue with 10 levels (0-9). Scheduling is done using RR at each level. Each queue is strictly FIFO (on moving to a queue or entering the system, a thread is added to the end of the queue). Threads at level \(n\) has absolute priority over those at level \(m\) if \(n < m\). The priority \(TP\) of the thread \(T\) is computed as follows:

(a) Each clock tick, if \(T\) was running, increment \(TP\) by one.
(b) Every 3 ticks, decay \(TP\) as follows: \(TP = \left\lfloor \frac{2^{\text{load}}}{\text{load} + 1} \right\rfloor T_P\), where load is the number of threads currently in the system. (All priorities are adjusted at time 3, 6, 9, etc.) For the thread that was running, \(TP\) is incremented before the decay takes place.
(c) If \(TP\) is greater than 9, thread \(T\) remains in the last queue of the system for scheduling.
(d) When \(T\) first enters the system, \(TP = 0\)

Show the gant chart for the execution of the following job sequence and the average response time, where response time is defined as time of completion minus arrival time. The quantum is 2 time units.

<table>
<thead>
<tr>
<th>Thread</th>
<th>Arrival Time</th>
<th>Computation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

5. [10pts] What is the major problem that the designers of BVT propose to address? What is the main idea of the solution? What makes the solution hard to implement in real life?

6. [10pts] In parallel processor scheduling, explain why, in most cases, space sharing (process control) gives better performance (in both average response time and system utilization) than time sharing (that is, gang scheduling). On the other hand, what situation can cause time sharing to perform better than space sharing?
7. [10pts] You are running one process on a machine with 3 frames of physical memory. The process emits the following virtual page reference string: 4, 0, 1, 2, 5, 5, 0, 2, 4, 2, 0, 1, 3, 2. Assume a virtual address space of 6 pages. Give the following: (i) number of page faults and (ii) the page table for the process if the replacement algorithm is:

(a) FCFS
(b) LRU
(c) 2nd-chance (or Clock). Assume that all reference bits are originally 0 and that the hand starts at frame 0.

8. [10pts] What is the major problem that the designers of Opal propose to address? What is the main idea of the solution?