Department of Computer Science
Operating Systems Qualifying Exam
August 18, 2004

The following exam is open book and open notes. You may feel free to use whatever additional reference material you wish, but **no electronic aids** are allowed. Please note the following instructions. There will be a ten point deduction for failure to comply with them:

- show your work whenever appropriate. There can be no partial credit unless I see how answers were derived.
- be succinct. You may lose points for facts that, while true, are not relevant to the question at hand
- Space is provided on this exam for solutions of some of the problems. You may either answer those problem on the exam or in your Blue Book.

1. (35 points) In order to focus on the issues of mutual exclusion, the following code is deliberately simplified in the following ways:

   - No memory management occurs within the code given here. The `insert()` function assumes the node to be inserted in the list is allocated separately, and given to it as a parameter. The `delete()` function returns the deleted node to the calling procedure.
   - Lists are always Long Enough. The search function always succeeds in finding the key in the list, and there is always a node in the list following the parameter given to `delete()`.
   - The nodes being manipulated are in shared memory, but any local variables in the functions are also local to the processor.
   - The memory uses serial consistency.

   A linked list is to be maintained in shared memory. A `Node` in the list is declared as

   ```
   typedef struct node *Node;
   struct node {
       int searchkey;
       Lock lock;
       Node next;
   }
   ```

   The code to search for a `Node` with a given `searchkey` is:

   ```
   Node search (Node list, int searchkey) {
       Node ptr;
       for (ptr = list; ptr->searchkey != searchkey; ptr = ptr->next);
       return ptr;
   }
   ```

   The code to insert a new `Node` in the list (*after* the `Node` pointed to by `listloc`) is:

   ```
   void insert(Node listloc, Node newnode, int searchkey)
   {
       newnode->next = listloc->next;
       listloc->next = newnode;
   }
   ```

The code to delete a `Node` from the list (*after* the `Node` pointed to by `listloc`) is:

```
Node delete(Node listloc)
{
    Node tmp;
    tmp = listloc->next;
    listloc->next = tmp->next;
    tmp->next = NULL;
    return tmp;
}
```

(a) Show that if two `insert()` operations occur simultaneously, it is possible for only one node to actually be inserted in the list.

(b) Show that if a `search()` operation occurs simultaneously with a `delete()` operation, it is possible for the `search()` operation to dereference a `NULL` pointer (even though it is not at the end of the list).

Now, assume you have operations called `Readlock(Lock lock)`, `Writelock(Lock lock)`, and `Unlock(Lock lock)`. As you might expect, these are atomic operations which manipulate `Lock`s according to the normal semantics: a `READ Lock` can be acquired if no process has a `WRITE` lock on the `Lock`, a `WRITE Lock` can only be acquired if no process has either a `READ` or a `WRITE Lock`. If a requested `LOCK` cannot be acquired immediately, the process waits until it can be acquired.

In the following, at the end of the (modified) functions all `Lock`s must be released. Also, the less time a `Lock` is held the better your answer is. You may also perform any code transformation you like that don't affect the semantics of the functions (for instance, replacing `for` loops with `while` loops, or introducing local variables) in order to place locks in the code as needed.

(c) Add `Lock` operations to the `insert()` function to guarantee it operates correctly in the situation in Question 1a

(d) Add `Lock` operations to the `search()` and `delete()` functions to guarantee they interoperate properly.

2. (25 points) Suppose you have a computer system running a mix of multimedia and hard real-time programs.

The real-time tasks have the following arrival times, CPU requirements, and deadlines, measured in "time slices."

Process	Arrival Time	CPU Requirements	Deadline
P_1	1	4	10
P_2	4	5	15
P_3	11	3	17
P_4	15	2	20

There is one multimedia process, P_M,which requires two time slices out of every five-slice epoch. If it isn't possible to schedule the multimedia task for two slices in an epoch then it isn't able to accomplish that epoch's processing, so there is no point in giving it any time in that epoch at all. The real-time processes have a higher priority than the multimedia process, in the sense that they should be scheduled to complete by their deadlines even at the expense of the multimedia task missing an epoch.

Schedule the five processes P_1through P_4 and P_M so that each real-time process meets its deadline and the multimedia task misses as few epochs as possible.

Process	Time																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P_1																				
P_2																				
P_3																				
P_4																				
P_M																				

3. (20 points) A four processor distributed-memory computer executes the code shown in the following figure. Note that there is no synchonization, so the operations may be performed with any interleaving. The initial value of x is 0, and it is homed on processor P_1.

P_1	P_2	P_3	P_4
`x = x + 1;`	`x = x + 4;`	`x = x + 2;`	`x = x - 3;`

   (a) Using the notation in which *`op(variable)value`* denotes a memory operation (as in `R(z)8` would mean "the processor reads variable `z` and gets the value `8`), show an interleaving for this code in which the final value of `x` is `0`. Assuming the variable is homed on processor P_1, and an Alewife-style communication is used, draw arrows showing the communication going between the processors.

P_1	
P_2	
P_3	
P_4	

   (b) Insert syncronization operators S in the code which will guarantee that the result is 4.

4. (5 points) On my computers at home, I use RSA-based authentication to let the root user on one machine access the other machines without a password. The public key for the root user on one of my computers is

```
AAAAB3NzaC1yc2EAAAABIwAAAIEAuO1epmzPt5JHTpiL4oJZEbVkm7I9SYl
1pbM1iWR0En0GqRqY10N2ERLIEJw77Skrx9MoTkp74aSabPpyqNk1tD2aFG
KuYmnyCTfEnR1xMH9bsCWFhUrmLLPQG/XcIpEp14x2hHyFtfXQ8YGWWhR6S
Cqq2O2J82uS7k07h0wt4VM=
```

   Why can I be confident that I haven't just given you any information which would help you actually break into my home computers? Answer in terms of both information I didn't give you, and how hard it would be to obtain that information.

5. (15 points) A computer with a virtual memory generates the following virtual page reference string:

   `1 2 3 4 2 6 2 3 7 1`

   Assuming your physical memory can hold three pages at a time, which pages will be in memory following each reference using each of the following page replacement strategies?

   (a) FIFO

1	2	3	4	2	6	2	3	7	1

   (b) LRU

1	2	3	4	2	6	2	3	7	1

   (c) Optimal

1	2	3	4	2	6	2	3	7	1