DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033—Computer System Engineering Handout 11—March 10, 1995

Quiz #1

This quiz contains five problems whose relative weights are indicated below. Please place each answer on the page with the question and hand in this booklet at the end of the examination.

Put your name on this cover sheet AND at the bottom of each page of this booklet. The booklet may be separated for grading so that each problem can be evaluated by one grader. Be sure that you don't put part of the answer to a problem on the back of a sheet for another problem!

Some parts of some problems may be much harder than others. Read them all through first and attack them in the order that allows you to make the most progress. Write down any assumptions and show your work, or you risk losing partial credit. Be neat. If we can't figure out your answer we can't give you credit. On answers that involve numbers, be sure to specify clearly the units of your answer.

Circle your recitation instance:

1. 10:00—Dally/Lew
7. 11:00—Dally/Philips
5. 11:00—Corbató/Dampier
2. 11:00—Hawley/Lew
6. 12:00—Hawley/Dampier
3. 1:00—Saltzer/Philips
8. 1:00—Corbató/Mitra
9. 1:00—Bedichek/Shih
4. 2:00—Saltzer/Mitra

For Official Use Only

Problem	Your Score	Maximum Score
1		18
2		20
3		18
4		20
5		24
Total		100

Name: ____________________________________

1. *(18 points)* For each of the following statements, circle whether it is True or False, and briefly explain. An answer without an explanation will receive no credit. And if you give a good enough explanation you may be able to get credit even if you say False to a point we thought was True, or vice-versa. *But keep it brief!*

1a. The paper "On building systems that will Fail", written by Corbató, was on creating planned catastrophes. TRUE/FALSE

1b. If one created a graph of personal friendships, one would have a hierarchy. TRUE/FALSE

1c. One of the mistakes that the Therac-25 programmers made was to not reuse software from earlier radiation therapy machines. TRUE/FALSE

1d. Firewalls completely prevent any bug in a server from affecting clients. TRUE/FALSE

1e. Modularity and abstraction control propagation of effects.TRUE/FALSE

1f. Variable delay is an intrinsic problem of isochronous networks.TRUE/FALSE

Name: ____________________________________

2. Ben Bitdittle is designing a virtual memory system, VM, for his new microkernel system Pico. In particular, he is interested in what page size to use for his VM system and whether VM should be implemented inside the Pico kernel or outside.

2a. *(5 points)* Ben's computer has a disk, which can be accessed in 20 msec. Once accessed Ben's computer can blast data to the disk at 10 Mbyte/s. Give a formula for the time to write an *n*-byte page to disk, in milliseconds.

2b. *(5 points)*Using the formula compute how long it takes to write a 1-Kbyte page and how long it takes to write an 10-Kbyte page to disk.

The time for a 1-Kbyte page =

The time for a 10-Kbyte page =

2c. *(5 points)*Given the results of b) Ben decides to use a 10-Kbyte page size.Why might Ben be better off choosing a smaller page size?

2d. *(5 points)*Suppose that crossing the kernel boundary and delivering a message takes about 100 cycles on Ben's 200-Mhz computer. Should Ben handle page faults in the kernel or in a trusted server?

Name: ________________________________

3. Two threads, A and B, execute a function with the following code (using the syntax from Birrell's thread paper), but always at different times (i.e., only one of the threads is in the function at a given time).

```
lock mutex_a do
    lock mutex_b do
    ...
    end
end

lock mutex_b do
    lock mutex_a do
    ...
    end
end
```

3a. *(9 points)*Assuming that no other code in the program ever acquires more than one lock at a time, can the program deadlock? (if yes, give an example, if not, argue why not)

3b. *(9 points)*Now, assume that the two threads *can* be in the code fragment above at the same time, can the program deadlock? (if yes, give an example, if not, argue why not)

Name: ________________________________

4. Threads in a new multithreaded WWW browser periodically query a nearby WWW server to retrieve documents. On average, a browser's thread performs a query every N instructions. Each request to the server incurs an average latency of T milliseconds before the answer is received.

4a. *(5 points)*For N = 2,000 instructions and T = 1 msec, what is the smallest number of such threads that would be required to keep a single 100 MIPS processor (i.e., executing one hundred million instructions every second) 100% busy? Assume the context switch between threads is instantaneous, and the scheduler is optimal.

4b. *(5 points)*But context switches are not instantaneous. Assume that a context switch takes C instructions to perform. Recompute the answer to part a) for C = 500 instructions.

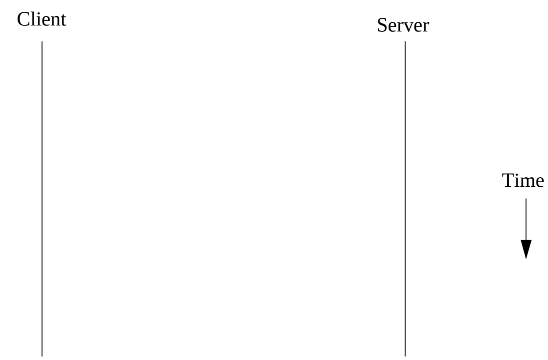
4c. *(5 points)*What property of the application threads might cause the answers of parts a) and b) to be incorrect? That is, why might more threads be required to keep the processor running the browser busy?

4d. *(5 points)*What property of the actual computer system might make the answers of a) and b) gross overestimates?

Name: ______________________________

5. Birrell and Nelson's RPC, as described in their paper "Implementing Remote Procedure Calls", guarantees *at-most-once* semantics (*i.e.*, the RPC is carried out at most one time, but possibly none at all).

5a. *(8 points)*Give a scenario (*i.e.*, a sequence of events) in which the same request may arrive twice at the server?



5b. *(8 points)*Ben gets really excited about RPC and implements an RPC package himself. Like Birrell and Nelson, he starts a thread for each new request at the server. The thread performs the operaton specified in the request, sends a reply, and terminates. After measuring the RPC performance, Ben discovers that the creation of the thread and context switching to it is a major part of the cost of performing an RPC. After thinking about it, Ben comes up with a brilliant optimization: instead of creating a new thread for each new request, Ben's optimized RPC package executes the RPC on the current running thread. To his surprise the server deadlocks with this new optimization. What is happening? (Hint: think about locks that the current thread might be holding.)

Name: ______________________________

5c. *(8 points)*Frustrated with optimizing his RPC package, Ben decides to use a brute force solution: he buys a much faster network. The latency for a sending packet from A to B on the new network is twice as low. Ben measures the performance of small RPCs (*i.e.,* each RPC contains only a couple of bytes) on the new network. To his surprise the performance is hardly better. What might be the reason that his RPCs are not twice as fast?

Name: ________________________________