Stanford University
Computer Science Department
CS 240 Quiz 1

April 30, 2008

This is an open-book exam. You have 50 minutes to answer **six of eight short questions** and **two of three long questions**. Write all of your answers directly on the paper. Make your answers as concise as possible. Sentence fragments ok.

NOTE: We will take off points if a correct answer also includes incorrect or irrelevant information. (I.e., don't put in everything you know in hopes of saying the correct buzzword.)

Question	Score
1-8 (30 points)	
9-11 (20 points)	
total	

Stanford University Honor Code

In accordance with both the letter and the spirit of the Honor Code, I did not cheat on this exam nor will I assist someone else cheating.

Name and Stanford ID:

Signature:

Answer six of the following eight questions and, in a sentence or two, *say why your answer holds.* (5 points each).

1. Your MESA code has two monitors, M1:

   ```
   entry foo1() { bar1(); }
   entry foo2() { bar2(); }
   ```

   And M2:

   ```
   condition c;
   entry bar1() { wait(c); }
   entry bar2() { signal(c); } }
   ```

   What can happen? Why does MESA work this way compared to the alternative?

2. Consider the manual/automatic stack management and serial/cooperative/preemptive labels in Figure 1 in the Cooperative Threading paper by Adya et al. How would you label:

   (a) The threaded programs checked by Eraser.

   (b) A Mesa program.

   (c) The commands "`emacs foo.c; gcc foo.c`"

3. Assume that on your machine a read always returns the value of the last write. If you compile and run the code below using the compiler in the Boehm paper:

```
          % initial: a = 0, b = 1
        thread 1   |    thread 2
---------------|---------------------
  r1 = a;      |   r3 = b;
  r2 = a;      |   a = r3;
 if(r1 == r2)  |
       b = 2;  |
---------------|------------------
```

Give the smallest and largest number of registers `r1`, `r2`, and `r3` that could hold the value "2" after this code runs (and explain how).

4. BVT: re-draw Figure 1 for an idealized scheduling system that has no error in its proportional sharing. Assume you have 1000 processes, each with one share apiece, explain how they will be scheduled and when the worst error will occur.

5. You access a page not in your resident set on VMS — give the minimum and maximum number of pages it could be *required* to read from disk and why you decided on these numbers (Hint: max is at least two.)

6. The VMS guys want to adapt ESX's proportional share to improve VMS's resident set limit. While they adore ESX, they hate periodic sampling so want to use either page rescues or resident set evictions to estimate idleness — how would you do so and which is better? (Hint: think about big arrays.)

7. What are two reasons that a superpage system could perform worse than a non-superpage system despite the extra-ordinary measures the superpage implementors took to guard against this?

8. You run the `matrix` benchmark in the superpage paper on the "Comparison of virtualization..." system with the important change that you use very small matrices. Give one argument *each* for why you may expect: (1) software virtualization to outperform the hardware approach and (2) there to be no difference.

Problem 9 You are looking at code in the Adya et al "Cooperative Task Management" system and see the routine:

```
void equal(CAID ca1, CAID ca2) {
      return GetCAInfo(ca1) == GetCAInfo(ca2);
}
```

To show off you decide to rewrite the code to use `GetCAInfoHandler1` instead of `GetCAInfo`, in such a way as to maximize concurrent I/O requests. Sketch how the code looks. (You needn't give every semi-colon but there should be enough detail to distinguish your answer from someone who doesn't understand events and continuations.)

Problem 10 Assume you have a runtime routine, `reachable(v)`, that Eraser can call that magically tells it there was any possible execution of the program that would allow a different thread to concurrently observe the memory associated with `v`.

(a) Explain how to adapt Eraser's state diagram (Figure 4) to use `reachable`: please be very concrete about any modifications you make with respect to error transitions or lockset refinements.

(b) Give an example false positive from the case studies that your brave new algorithm would eliminate.

(c) Assume `reachable(v)` instead just indicates if a thread *at this moment* can read `v` — does this change things in any significant way?

Problem 11 Suppose you have three virtual machines, A, B, and C, running on top of VMWare ESX. Each configured so the guest OS sees 100 MB of memory (meaning the VM's *max size* parameter is 100 MB). There are 180 MB of physical machine memory available to partition among these virtual machines after VMware's own data structures. All three machines run different guest OSes and experience minimal page sharing.

Machine A is more important than the other two machines, and is configured with $S = 4$ shares for memory allocation, while B and C have only one share each. On boot-up, all three machines touch all 100 MB of memory, but then A and B become completely idle, while machine C continues to use all of its memory. Assume the idle memory tax $\tau = 75\%$. How much memory will end up allocated to virtual machine A? Explain your reasoning.