CS 162 Midterm — March 17, 1997

Your name ____

login cs162–_______

Discussion section number _______

TA's name ____

This exam is worth 20 points, or 20% of your total course grade. The exam contains seven questions.

This booklet contains six numbered pages including the cover page. Put all answers on these pages, please; don't hand in stray pieces of paper. This is an open book exam.

1	/2
2	/2
3	/2
4	/6
5	/3
6	/2
7	/3
total	/20

Question 1 (2 points):

In project 1 we told you that you didn't have to turn off interrupts in your implementation of semaphores, but in project 2 we told you that you *did* have to turn off interrupts in parts of your scheduler. **In no more than three sentences,** explain this difference.

Question 2 (2 points):

On page 66, Tanenbaum describes a bug in the CTSS scheduler: To aid interactive jobs, CTSS increased the priority of a process whenever the user completed a line. This feature could be abused by a compute-bound user who would keep typing carriage returns every few seconds.

In no more than two sentences, how should this bug be fixed, while maintaining the idea of higher priority for interactive jobs?

Question 3 (2 points):

Louis Reasoner says, "the `rename()` system call in Unix is confusing because it works only if the new name will be in the same file system as the old one.." (This is true.) "The system's behavior will be more uniform if instead of a `rename` operation, we rename files by copying the old file to the new name, and then deleting the old file." Leaving out the inefficiency of doing extra disk operations, explain **in no more than two sentences** how Louis' proposal would change the behavior of Unix in a user-visible way.

Your name ______________________________ login cs162–________

Question 4 (6 points):

Reproduced here are two students' solutions to the bridge traffic problem from our homework. For each of these solutions, indicate whether it is correct or incorrect, and, if incorrect, under what circumstances it fails. **If a solution is incorrect, show how to fix it.** Both of these solutions use the monitor abstraction; don't worry about the notation.

Solution A:

```
monitor ArriveExit {
    int direction = 0, cars = 0;
    condition arrive;

    void ArriveBridge(int direc) {
        while (cars >= 3 || ((direc != direction) && cars != 0))
            wait(arrive);
        cars++;
        direction = direc;
    }

    void ExitBridge() {
        cars--;
        signal(arrive);
    }
}
```

Problem continues on next page.

Question 4 continued:

Solution B:

```
monitor ArriveExit {
    condition safe, not_full;
    int carCount = 0, direction = 0;

    void ArriveBridge(int direc) {
        if (carCount == 0)
            direction = direc;
        while (direction != direc && carCount != 0)
            wait(safe);
        direction = direc;         /* change direction */
        carCount = carCount + 1;
        while (carCount >= 3)
            wait(not_full);
    }

    void ExitBridge() {
        carCount = carCount - 1;
        if (carCount == 0)
            broadcast(safe);
        if (carCount < 3)
            broadcast(not_full);
    }
}
```

Your name ____ login cs162–________

Question 5 (3 points):

This question refers to an architecture using segmentation with paging. In this architecture, the 32-bit virtual address is divided into fields as follows:

```
+----------------------+--------------------+----------------+
| 4 bit segment number | 12 bit page number | 16 bit offset |
+----------------------+--------------------+----------------+
```

Here are the relevant tables (all values in hexadecimal):

```
Segment Table               Page Table A             Page Table B

 +-----------------+         +-----------------+       +-----------------+
0| Page Table A    |        0| CAFE            |      0| F000            |
1| Page Table B    |        1| DEAD            |      1| D8BF            |
x| (rest invalid) |        2| BABE            |      x| (rest invalid) |
 +-----------------+        3| BEEF            |       +-----------------+
                            x| (rest invalid) |
                             +-----------------+
```

Find the physical address corresponding to each of the following virtual addresses (answer "bad virtual address" if the virtual address is invalid):

a. 00000000

b. 20022002

c. 10015555

Question 6 (2 points):

In some operating systems, I/O from/to disk is done directly to/from a buffer in the user program's memory. The user program does a system call specifying the address and length of the buffer. (The length must be a multiple of the disk record size.)

The disk controller needs a physical memory address, not a virtual address. So, Ben Bitdiddle proposes that when the user does a `write` system call, the operating system should check that the user's virtual address is valid, translate it to a physical address, and pass that address and the length (also checked for validity) to the disk hardware.

This won't quite work. **In no more than two sentences,** what did Ben forget?

Question 7 (3 points):

In no more than three sentences, what will be the long-term effect if the lottery scheduler mistakenly gives a process that blocked before completing its time slice too many compensation tickets once?