NAME: ______________________________________ (FIRST NAME FIRST) **SCORE**: ____

COSC 4330 **FIRST MIDTERM** **OCTOBER 2, 2002**

This exam is **closed book**. You can have **one** page of notes.

1. *Explaining why* (5 points each).

a) Why do most operating systems on the market continue to use ***monolithic kernels***?

Because monolithic kernels are faster than microkernels.

b) Why are ***timer interrupts*** so important?

Because they are needed to prevent a running process doing no I/O operations from keeping the CPU forever.

c) Why should we ***prevent*** users of a multi-user system from ***rebooting the OS from a floppy disk***?

Because they could reboot the system with a doctored kernel allowing them unrestricted access to other users' files.

d) Why was MS-DOS ***inherently insecure***?

Because it gave user processes direct access to the BIOS I/O routines.

e) Why are ***layered kernel organizations*** impractical?

Because the multiple functions on an OS are highly interdependent and cannot be easily layered one on the top of another.

f) Why will we never see hard drives with access times ***below one millisecond***?

Because they would have to spin too fast.

2. Somebody proposes to you replace the **fork()** and **execvp()** system calls of UNIX by a single system call, say, **newprocp(filename, argv)** combining the functions of **fork()** and **execvp()** in a single system call.

a) What would be the ***major advantage*** of this solution? (5 points)

It would combine two system calls into one and eliminate the cost of having the fork() making a copy of the data segment of the parent process in the address space of the new process.

b) Which features of UNIX would stop working if the **fork()** and **execvp()** system calls were removed ? (2×5 points)

I/O redirection and pipes.

3. How many lines will the following program print out? (5 points)

```
main() {
     fork();
```

```
        printf("Hello!\n");
        fork();
        printf("Goodbye!\n");
}
```

Answer: The program will print out exactly __6__ lines.

4. Compare and contrast the ***master-slave*** and the ***symmetric*** approaches to building multiprocessor operating systems. (10 points)

The master/slave approach requires less changes to the kernel but creates a potential bottleneck as there is a single copy of the kernel that has to respond to the requests of all user processes.

5. Represent all the possible transitions among the ***ready***, ***running*** and ***waiting*** states (6 points) and describe in one or two sentences those involving the ***ready*** state (9 points). ***Note****: You will lose points if you represent other states or other transitions than those requested.*

(see class notes).

6. How will the following code fragment affect **stdin, stdout and stderr**? (5 points each)

```
int fd, pd[2];
pipe(pd);
fd = open("data.txt", O_RDWR | O_CREAT, 0640);
close (0);
dup(fd);
close(1);
dup(pd[1]);
```

stdin is reading now from the file data.txt ________________________________

stdout is redirected to the pipe pd ____

stderr is unchanged ____

7. You have the choice between writing a program using user-level threads or kernel supported threads. Which factors should affect your decision when both types of threads are supported by the OS on which the program will run? (2×5 points)

User level threads are faster but require the use of non-blocking I/O calls.