CSSE 332 – Operating Systems
Rose-Hulman Institute of Technology
Computer Science and Software Engineering Department

Exam 1 – Paper Part

Name:_____ Section:_____ CM:_____

This exam consists of two parts. The first part is to be done on paper without using your computer. The second part of the exam is to be done on your computer. You have the full lab period (a total of three periods) to complete the entire exam.

Instructions: The paper part of the exam is closed book, open notes limited to one double-sided sheet of hand written notes, but **no computer or electronic devices**. Write all of your answers in the spaces provided.

When you complete the paper part of the exam, turn it in to an exam proctor. You may then begin using your computer. **Use of your computer before turning in the paper part of the exam will be considered academic dishonesty.**

To allow sufficient time to work on the computer part of the exam, we suggest using no more than 50 minutes for the paper part.

Please begin by putting your name on the first page and your initials on every page of the exam. We encourage you to skim the entire exam before answering any questions and show all your work to receive partial credit.

	Points available	Your Points
1	10	
2	5	
3	5	
4	3	
5	3	
6	2	
7	2	
Total	100	

Problem 1 (10 points) Fill in the Gantt charts, using the given scheduling policy and the table below. If a new process is admitted at the same time that an existing process is preempted, the new process enters the ready queue first.

Note that all processes will not finish in the given slots! Just fill up the 10 slots provided and move to the next question.

Note: processes are always enqueued at the tail of the ready queue and dequeued from the head of the queue.

Process	Arrival Time	Service Time
A	0	3
B	1	6
C	2	5
D	3	1

	0	1	2	3	4	5	6	7	8	9
Round robin, quantum = 3										
Shortest job first (SJF)										

Problem 2 (5 points) Imagine a system is using round robin scheduling. What happens when a cpu-bound process had used the max quantium time? In particular: How does the OS regain control? What state does the cpu-bound process go into now that it's no longer running?

Problem 3 (5 points) Imagine a process is running on a CPU when it makes a blocking IO call. Explain all the states the process from that moment till it is in the running state again with the results of it's IO call. Be sure to include: the names of each state, what triggers the process to enter that state.

Problem 4 (3 points) Imagine you're using an OS with a Many-to-1 model of threading. That is, all a single process's user threads map to a single kernel thread. Now one user thread makes a blocking IO call. What happens to the other threads in that process? To the threads in other processes?

Problem 5 (3 points)

```
1   int fd[2];
2   FILE* pipestream;
3   pid_t result;
4   int compute;
5
6   pipe(fd);
7   result = fork();
8   if(result == 0) {
9     close(fd[0]);
10    pipestream = fdopen(fd[1], "w");
11
12    compute = someFunction();
13    fprintf(pipestream,"%d\n", compute);
14    return 0;
15  }
16  // other code...
```

Briefly explain what this code is doing.

Problem 6 (2 points)

Recall this equation for estimating how long a process will run before blocking:

$$\tau_{n+1} = \alpha t_n + (1 - \alpha)\tau_n$$

If alpha is selected to be near one, what suggest about the assumptions the OS is making about process behavior?

Problem 7 (2 points) Imagine you're writing a program that's designed to process data in parallel. You read in a datafile, each line of which gives data for a particular computation. For each line, you start a new thread to process that particular line. The file can be of arbitrary size. Why is this a bad idea?