Name_____SAMPLE_____

Last 4 digits of Student ID_____

CSE 3320 GRADE:_____/100

Operating Systems

Exam 2, Fall 2012 (Non scary, Halloween edition)



**Please** read this:
This is a closed book, closed notes exam. You may, however, use one sheet of notes. Use a dark ink (or pencil) and print answers on the test paper following the question or on separate paper. Please **do not write on the back of sheets** of paper **or** on the **back of the exam**. Please **keep answers to questions together** and try to avoid continuing answers off the page. Please answer the questions in a **few keywords**, complete sentences are not necessary, **be brief**! Please put your name on each page. Turn in all pages of the test. **Write** your answers **legibly**. Unreadable answers will be counted wrong. Make sure you have all pages of the test.
**Read** each question **carefully** (twice) and be sure your answer addresses the question. Overly general (non-specific) answers will be counted wrong. If any part of an answer is incorrect (even if other parts are correct) points will be deducted.
Point values are given for each question. The exam has a total of 100 points.
Please turn in the exam promptly when called for. **Late exams will have points deducted**.

**"Explain" - means briefly describe why - just a few words.**

**You do not need to calculate the final result - just show the calculation, and your reasoning.**

1. [25 pts] Short Answer:
   (a) For two of the more unusual aspects of Unix (compared to previous OSs):
       (1) What programming language was used to write (version 6 and newer) Unix? What programming language was used to write most other OSs at that time?
       (2) The file system seems to contain elements that are not actually files, Please give an example of one of these. What benefit does this give to programmers?
   (b) MacOS (System 4) used cooperative multitasking and Unix/Linux used preemptive tasking to allow multiple jobs to run.
       (1) Briefly, how do these approaches differ?
       (2) What is one advantage of each method?
   (c) In a 12 year time span, Apple released 9 versions of MacOS (system 1 through 9). Why? Why not one, single (possibly configurable) version like IBM and CP/M did?

2. [40 pts]
The following is a list of processes:

| Process # | Arrival Time | Memory Size | Running Time | Priority |
|---|---|---|---|---|
| 1 | 0 | 20K | 12 | 5 |
| 2 | 0 | 100K | 2 | 4 |
| 3 | 6 | 100K | 4 | 1 |
| 4 | 8 | 10K | 10 | 4 |
| 5 | 14 | 20K | 12 | 1 |
| 6 | 14 | 50K | 5 | 2 |
| 7 | 14 | 200K | 1 | 1 |
| 8 | 24 | 10K | 4 | 2 |

Where arrival time and running time are in seconds, and memory size in K Bytes.
If appropriate the time quantum = 2 second. Lower Priority values are higher priority.
(When arrival time is the same, select lowest process number first, if more than one process can be selected use "most fair" - allow next (newest) process to run next.
Please do not "combine" scheduling policies, unless scheduler normally does that.)

(a) (1) Please show the CPU timeline (GANTT chart) for a FCFS scheduler.
    (2) What is the turnaround time for Job 2 and Job 7?
(b) (1) Please show the GANTT chart for a Shortest Job Next (SJN) scheduler (with no preemption).
    (2) What is the wait time (time in wait queue) for Job 2 and Job 6?
(c) Please show the GANTT chart for a SJN scheduler, with preemption.
(d) Please show the GANTT chart for a Round Robin (RR) scheduler.
(e) Please show the GANTT chart for a Priority scheduler, with no preemption.
(f) Please show the GANTT chart for a Priority scheduler, with preemption.
(g) Comparing: 1. FCFS, 2. SJN with preemption, 3. RR and 4. Priority with preemption
    (1) Which scheduler has the minimum turnaround time for Jobs 4 and 5?
    (2) Which scheduler has the worst turnaround time for Jobs 4 and 5?
    (3) Which scheduler has the best throughput?

3. [35 pts]
There are two processes running on two cores in one CPU (same OS).
They are sharing a memory location called "share" (with a start value of 0).
P1: while(1) { temp = share; temp = temp + 1; share = temp; }
P2: while(1) { temp = share; cout << temp; }

(a) Will this print the numbers: 1,2,3,4,5 etc consecutively with no gaps? Please explain.
(b) Using a lock or semaphore called "e" (exclude), modify P1 and P2 to be synchronized.
(c) (1) Show how, using the machine instructions "TestAndSet" or "Swap" you can implement a lock ("lock(e)")
    (2) What does lock(e) do, what does it return?
(d) With your part b modifications, can P1 and P2 deadlock? How or why not?
(e) Would lock(e) be part of the C library or the OS? Why?
    Would calling lock(e) be privileged? Why?

Bonus [5 pts]
Please explain how the Java JVM can ensure that threads sharing an object can have exclusive access. Show (briefly) how a Java program does this and explain (briefly) how the JVM implements exclusion.