Full Name: ______________________

CS 450 Summer 2009
Midterm Exam

July 6th, 2009

Instructions:

- This exam is closed-book, closed-notes.
- Keep your written answers concise and to-the-point. I reserve the right to deduct points for needless verbiage.
- Write your full name on the front, and make sure that your exam is not missing any sheets.
- Good luck!

Problem 1 (/12) :	
Problem 2 (/6) :	
Problem 3 (/6) :	
Problem 4 (/12) :	
Problem 5 (/10) :	
Problem 6 (/10) :	
TOTAL (/56) :	

Problem 1. (12 points):

Complete the process state-transition diagram below by adding and labeling directed edges between the states indicating all possible kernel-implemented state transitions. While you need not label all the graph edges you draw, at a minimum you should indicate those transitions that correspond to:

a. Preemption

b. I/O request / Interrupt

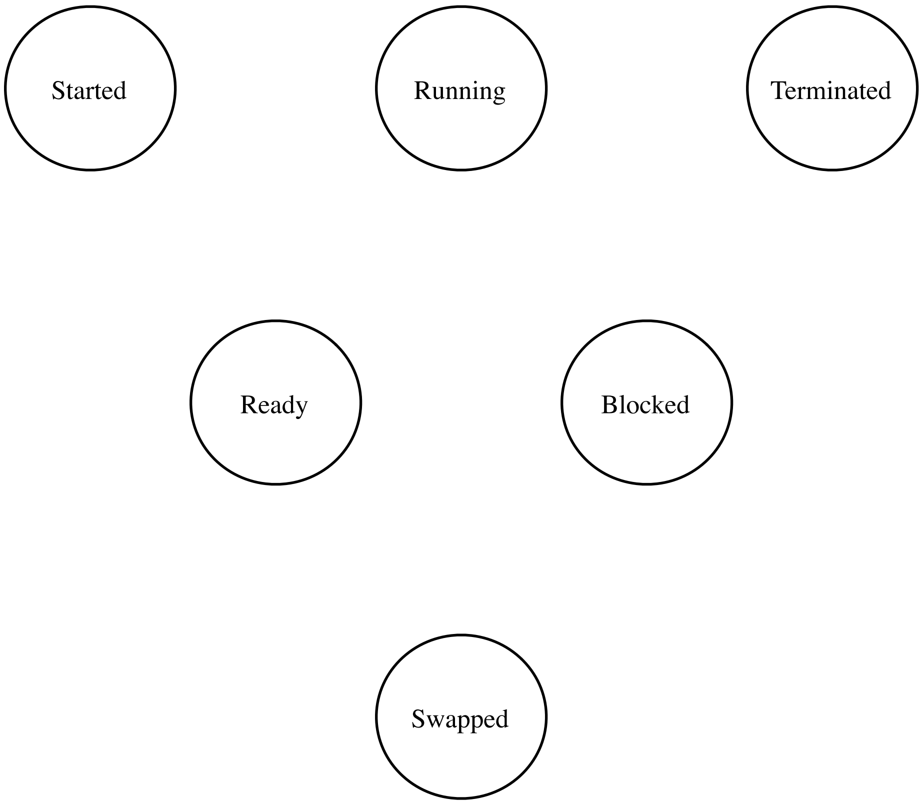
c. I/O request completion

d. Scheduler selection

e. Swap out (to disk)

f. Swap in (to core)

You may simply write the corresponding letter for each label given above next to the appropriate edge. Labels may be used more than once.



Problem 2. (6 points):

Is aging a sufficient mechanism for combatting priority inversion? Why or why not?

Problem 3. (6 points):

Consider the design of a multi-level feedback queue (MLFQ) scheduler consisting of a round robin queue with $q = 10$ms and a FCFS queue. 70% of the processor time is allocated to the RR scheduler, and 30% is dedicated to processes on the FCFS queue. Describe separate scenarios that may cause a given process to be moved from the RR to the FCFS queue and then back again.

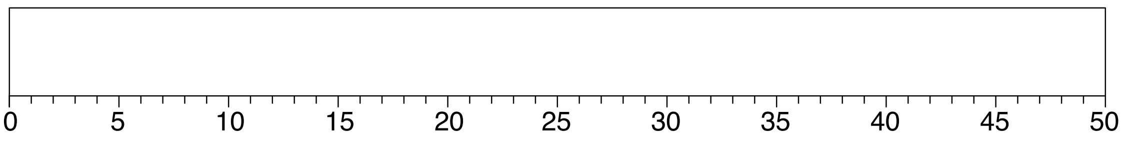
Problem 4. (12 points):

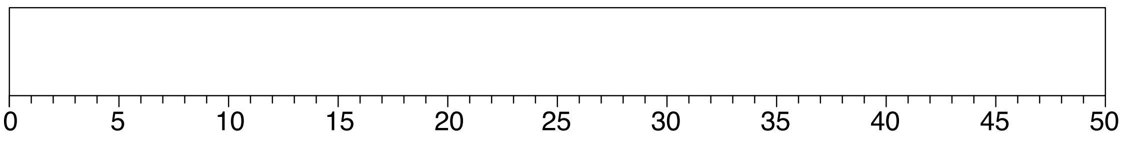
Consider the following two processes:

- P_1, arriving at time $t = 0$, completing after two CPU bursts of 10ms each separated by a single I/O burst of 15ms.
- P_2, arriving at time $t = 2$, completing after three CPU bursts of 5ms each separated by two I/O bursts of 10ms each.

Assume that there is no context switch overhead.

A. Fill in the following Gantt chart template to chart the execution of the two processes using the pre-emptive SJF scheduling algorithm. You should shade in sections where neither process is active. (Multiple templates are provided in case you mess up).





B. What is the total waiting time for each of the two processes? (Recall that waiting time does not include I/O overhead.)

C. What is the CPU utilization over the execution of the two processes – i.e., what is the fraction of time during which the CPU is not idle?

D. Could the CPU utilization be improved using a different scheduling algorithm? Justify your answer.

Problem 5. (10 points):

Consider the following resource snapshot for a system currently in deadlock:

	Allocated				*Requested*				*Available*			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	3	0	0	0	0	2	0	0	1	2	0	2
P_1	2	5	0	0	0	0	0	3				
P_2	0	0	2	0	4	0	0	0				
P_3	4	0	3	2	0	4	0	0				
P_4	0	1	5	0	0	0	2	2				
P_5	0	2	0	6	5	0	0	0				

A. What are the processes and resources involved in the deadlock?

B. How would you recommend we get out of the current deadlock? Be specific, and justify your answer.

Problem 6. (10 points):

The board of trustees has decided to install a number of lounges across campus for students and teachers alike to sit and relax in. The use of these lounges, however, is governed by a few rules:

- A lounge may be occupied solely by students or teachers – they may not intermingle!
- A maximum of 10 students may occupy a lounge at a time
- A maximum of 5 teachers (we're a little more stressed, see) may occupy a lounge at a time
- No single group may hog the lounge forever; i.e., if students currently occupy the lounge and students keep arriving to enter, teachers can't be kept out forever

For this problem you are to implement the code for the two types of threads that will be executed by individual students and teachers. All synchronization should be performed using the following variable definitions:

```
loungeEmpty = Semaphore(1)
turnstile   = Semaphore(1)

student_mtx = Semaphore(1)
teacher_mtx = Semaphore(1)

student_multiplex = Semaphore(10)
teacher_multiplex = Semaphore(5)

students = 0
teachers = 0
```

You should NOT introduce any other variables in your solution. Clearly indicate the separate threads and where each is free to "lounge". Use the empty space below or the following page for your solution.

You may use this page for your solution to problem 6.