

CS4411 Intro. to Operating Systems Exam 2 Fall 2005

150 points – 9 pages

Name: _____

- Most of the following questions only require short answers. Usually a few sentences would be sufficient. Please write to the point. If I don't understand what you are saying, I believe, for most of the cases, you don't understand the subject.
- *Justify your answer with a convincing argument.* If there is no justification when it is needed, you will receive no credit for that question even though you have provided a correct answer. *I consider a good and correct justification is more important than a right answer.*
- Problem(s) marked with an asterisk * are recycled problems. Grading policy for these recycled problems is *all-or-nothing*. More precisely, if you make any mistake, you will receive **no** credit.

1. Recycled Problems

- (a) [10 points]* Draw the state diagram of a process from its creation to termination, including all transitions, and briefly elaborate every state and every transition.

- (b) [5 points]* The methods `Wait()` and `Signal()` must be atomic to ensure a correct implementation of mutual exclusion. Use an execution sequence to show that if `Wait()` is not atomic then mutual exclusion cannot be maintained. **You must use an execution sequence as we did many times in class to present your answer. Otherwise, you risk low or no credit.**

2. Synchronization

- (a) [10 points] Enumerate and elaborate all major differences between a semaphore wait/signal and a condition variable wait/signal. Vague answers and/or inaccurate or missing elaboration receive no credit.

- (b) [5 points] Why is calling a monitor procedure from within another monitor (*i.e.*, nested monitor call) not a good practice?

3. Process Scheduling

- (a) [8 points] What are *preemptive* and *non-preemptive* scheduling policies? Elaborate your answer.

- (b) [8 points] We discussed in class that the shortest job first (SJF) scheduling policy is “*provably optimal*.” What does this mean? And, show that the SJF scheduling is optimal for three jobs based on your answer to the previous question. **Note that are two questions. You have answer each problem clearly and correctly.**

- (c) [20 points] Five processes *A*, *B*, *C*, *D* and *E* arrive in this order at the same time with the following CPU burst and priority values. A smaller value means a higher priority.

	<i>CPU Burst</i>	<i>Priority</i>
<i>A</i>	7	3
<i>B</i>	2	5
<i>C</i>	3	1
<i>D</i>	6	4
<i>E</i>	4	2

Fill the entries of the following table with waiting time and average waiting time for each

indicated scheduling policy and each process. Ignore context switching overhead.

<i>Scheduling Policy</i>	<i>Waiting Time</i>					<i>Average Waiting Time</i>
	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	
First-Come-First-Served						
Non-Preemptive Shortest-Job First						
Priority						
Round-Robin (time quantum=2)						

4. Deadlocks

(a) [8 points] What are the necessary conditions for a deadlock to occur? List these conditions and provide an elaboration. Providing conditions without an elaboration or providing a vague elaboration receives **no** credit.

(b) [10 points] Consider the following snapshot of a system:

	<i>Allocation</i>				<i>Max</i>				<i>Need</i>				<i>Available</i>			
	<i>U</i>	<i>V</i>	<i>W</i>	<i>X</i>	<i>U</i>	<i>V</i>	<i>W</i>	<i>X</i>	<i>U</i>	<i>V</i>	<i>W</i>	<i>X</i>	<i>U</i>	<i>V</i>	<i>W</i>	<i>X</i>
<i>A</i>	0	0	1	2	0	0	1	2	0	0	0	0	1	5	2	0
<i>B</i>	1	0	0	0	1	7	5	0	0	7	5	0				
<i>C</i>	1	3	5	4	2	3	5	6	1	0	0	2				
<i>D</i>	0	6	3	2	0	6	5	2	0	0	2	0				
<i>E</i>	0	0	1	4	0	6	5	6	0	6	4	2				

Is the system in a safe state? **Show your computation step-by-step; otherwise, you will receive no credit.**

(c) [8 points] Consider the following snapshot of a system:

	<i>Allocation</i>				<i>Request</i>				<i>Available</i>			
	<i>U</i>	<i>V</i>	<i>W</i>	<i>X</i>	<i>U</i>	<i>V</i>	<i>W</i>	<i>X</i>	<i>U</i>	<i>V</i>	<i>W</i>	<i>X</i>
<i>A</i>	0	0	1	0	2	0	0	1	2	1	0	0
<i>B</i>	2	0	0	1	1	0	1	0				
<i>C</i>	0	1	2	0	2	1	0	0				

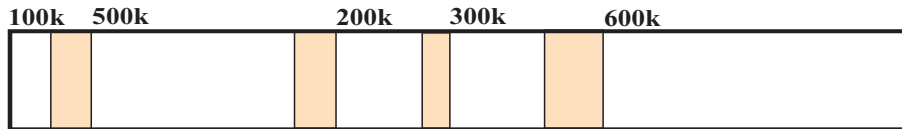
Is the system in a deadlock state? If the system is in a deadlock state, what are the processes that involve in a deadlock. **Show your computation step-by-step; otherwise, you will receive no credit.**

5. Memory Management

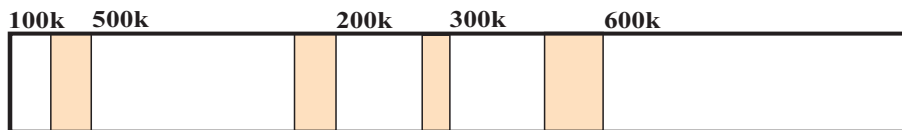
(a) [6 points] Define *external* and *internal* fragments. Which memory management scheme does not have external fragment? Why? **Note that there are three questions here.**

(b) [20 points] Given memory holes (*i.e.*, unused memory blocks) of 100K, 500K, 200K, 300K and 600K (in address order) as shown below, how would each of the *first-fit*, *next-fit*, *best-fit* and *worst-fit* algorithms allocate memory requests of 290K, 420K, 110K and 350K (in this order)? The shaded areas are used/allocated regions that are not available. Write your answer into the following figures. **Use shaded areas to indicate unused memory blocks.** You should write down the size of each allocated and unused memory block. Otherwise, you will receive **no** credit for that part.

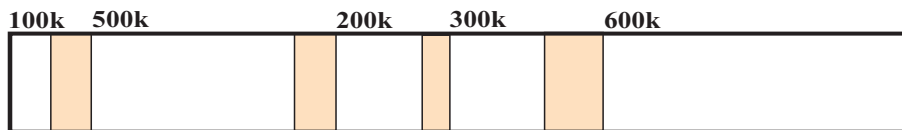
i. [5 points] **First-fit:**



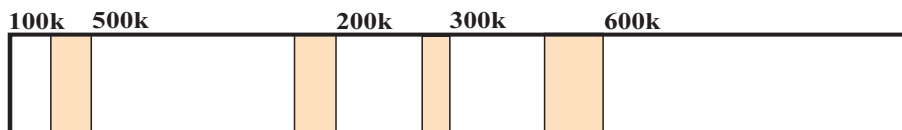
ii. [5 points] **Next-fit:** The current area is the 100K region.



iii. [5 points] **Best-fit:**



iv. [5 points] **Worst-fit:**



(c) [6 points] A paging system uses 16-bit address and 4K pages. The following shows the page tables of two running processes, Process 1 and Process 2. Translate the logical address 16,000 of Process 1 and the logical address 9,000 of Process 2 to their physical addresses. Fill your answers into the table below.

Process 1		Process 2	
0	0	0	3
1	4	1	1
2	5	2	7
3	2	3	6
		4	8

<i>Process</i>	<i>Address</i>	<i>Page #</i>	<i>Offset</i>	<i>Physical Address</i>
Process 1	16,000			
Process 2	9,000			

- (d) [6 points] Suppose both processes ask for a shared memory of 4K, and suppose further that the system decides to allocate page frame 10 for this purpose. What virtual (or logical) addresses processes Process 1 and Process 2 will receive and what are the new page tables? **You should provide sufficient reasoning. A simple answer will receive no credit.**

6. Programming [20 points]

Each thread in a system has a unique ID, which is a positive integer. The system also has a shared file that can be accessed by multiple threads simultaneously as long as the sum of the ID's of all threads that are currently accessing the file is less than a predefined value `MAXIMUM`.

Design a Hoare monitor `Strange` and monitor procedures `Access(id)` and `Release(id)`, where `id` is the ID of the calling thread. Monitor procedure `Access(id)` allows the caller to access the file if the sum of the all ID's and `id` is less than `MAXIMUM`. In this case, `Access(id)` returns. Otherwise, the caller is blocked. On the other hand, when a thread finishes accessing the shared file, it calls monitor procedure `Release(id)` to release the file.

Use `ThreadMentor` syntax to write the monitor code. It is required to have an explanation/elaboration of your design. Otherwise, you will risk low to very low grade. **Hint:** This problem looks easy; but, it has a tricky portion: what would happen if a blocked thread finds out it still cannot run after it is released? If you handle this situation improperly, the system may end up to have every thread blocked. This is the key issue when I will grade your paper.

Use the space on this page and the next blank page for your answer

Use this blank page for your answer

Grade Report

<i>Problem</i>		<i>Assigned</i>	<i>Received</i>
1	a	10	
	b	5	
2	a	10	
	b	5	
3	a	8	
	b	8	
	c	20	
4	a	8	
	b	10	
	c	8	
5	a	6	
	b(i)	5	
	b(ii)	5	
	b(iii)	5	
	b(iv)	5	
	c	6	
	d	6	
6		20	
Total		150	