CS4411 Intro. to Operating Systems Final
Fall 2009

200 points – 10 pages

Name: _____

- Most of the following questions only require very short answers. Usually a few sentences would be sufficient. Please write to the point, however. If I don't understand what you are saying, I believe, for most cases, you don't understand the subject well.
- **Justify your answer with a convincing argument**. If there is no justification when it is needed, you will receive no point for that question even though you have provided a correct answer. *I consider a good and correct justification more important than a right answer*.
- Repeated problems are marked with asterisks, and will be graded with the *all-or-nothing* policy. Unless your mistakes are very very minor, you receive no point.
- Except for the last two problems which are programming related, all problems test if you have understood the basic concepts and can be answered quickly. **Go through all problems once and do those you know how to do first.**
- The programming related problems are easy, if you understand the merit of channels and monitors.

1. **Threads and Synchronization**

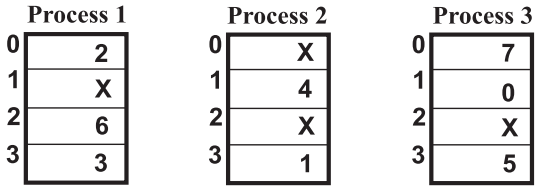
   (a) [**10 points**] Consider the solution to the mutual exclusion problem for two processes P_0 and P_1, where `flag[2]` is a Boolean array of two elements and `turn` is an integer variable with an initial value 0 or 1. Both are global to processes P_0 and P_1. Show, with a step-by-step execution sequence table, that mutual exclusion is implemented incorrectly. **You will risk low or very low grade if you do not use an execution sequence table.**

```
bool  flag[2];   // global flags
int   turn;      // global turn variable

Process i (i = 0 or 1)

flag[i] = TRUE;          // I am interested
while (turn != i) {      // while it is not my turn
     while (flag[j])     //   while you are interested
          ;              //     do nothing: busy waiting
     turn = i;           //   because your are not interested, it is my turn
}
// critical section
flag[i] = FALSE          // I am done and not interested
```

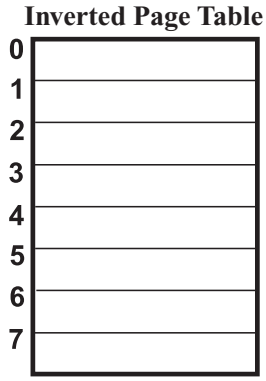
   (b) [**10 points**] A communication channel can be *synchronous* or *asynchronous*. What is the meaning of *synchronous* and *asynchronous*? Define these terms precisely.

2. **Memory Management**

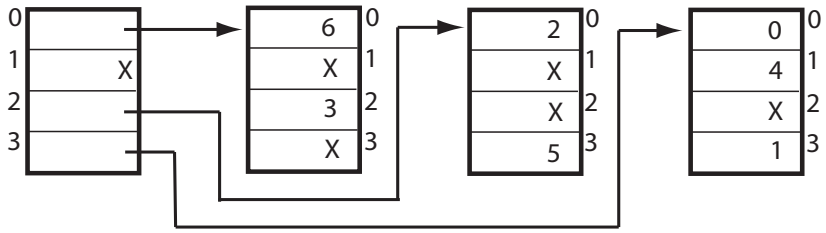
(a) **[8 points]** A system has 8 page frames and each process has a page table of four entries in which a "X" indicates the corresponding page is not in physical memory. Suppose a snapshot of the page tables at certain moment is shown below:



Convert the content of these three page tables to an inverted page table below.



(b) **[10 points]** Let page size be 4K. Each page table has four entries. Suppose we have the following two-level page tables in which only page frame numbers are shown, where an X means the corresponding entry is not in physical memory.



Translate the following virtual addresses to the corresponding physical addresses (in decimal). Fill in the table with your answers. Please only fill the table entries with **decimal** values. **Expressions are not acceptable.**

Virtual Addr.	*1st level page #*	*2nd level page #*	*Offset*	*Page frame #*	*Physical Addr.*
40,000					
30,000					

3. **Page Replacement Algorithms**

Let the page reference string be 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2. The computer on which this page reference string is run has three page frames. Run this page reference string with the FIFO, LRU and the Optimal algorithms. Initially, the memory is empty. Give the content *after* each reference and compute the number of page faults, miss ratio and hit ratio. **You should circle those pages that cause page faults**.

(a) **[10 points]** FIFO algorithm:

Page	2	3	2	1	5	2	4	5	3	2	5	2
Memory												
Page faults												
Miss ratio												
Hit ratio												

(b) **[10 points]** LRU algorithm:

Page	2	3	2	1	5	2	4	5	3	2	5	2
Memory												
Page faults												
Miss ratio												
Hit ratio												

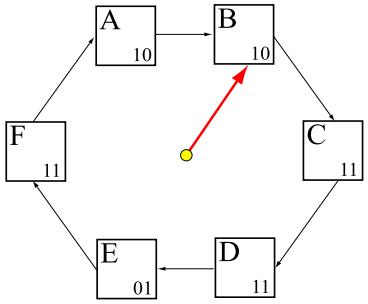
(c) **[10 points]** The Optimal algorithm:

Page	2	3	2	1	5	2	4	5	3	2	5	2
Memory												
Page faults												
Miss ratio												
Hit ratio												

(d) **[10 points]** Working set of window size $\theta = 3$:

Page	2	3	2	1	5	2	4	5	3	2	5	2
Memory												

(e) **[10 points]** Suppose the CLOCK page-replacement algorithm is used to approximate the LRU algorithm. Currently a system has six page frames with RM (Reference/Modified) bits as shown below (*i.e.*, the first and second bits are the reference and modified bits, respectively):



If the clock pointer is at the page frame as shown and a page fault occurs, after running the CLOCK algorithm, what are the new RM bits? Which page frame will be selected as a victim? Is a page-out required in this case? Why? Which page frame will be selected if a new page fault occurs immediately? **Note that there are <u>five</u> questions!**

(f) **[16 points]** Answer the following **<u>three</u>** questions. **(1)** What is Belady's anomaly ([**4 points**])? **(2)** What is the *inclusion property*([**4 points**])? **(3)** Show that a page replacement algorithm that satisfies the inclusion property does not have Belady anomaly ([**8 points**]).

(g) **[10 points]** What is *thrashing*? How can it happen? How can it be overcome? **Note that there are <u>three</u> questions. Elaborate your answer. Otherwise, you will receive <u>no</u> credit.**

4. **File System and Storage Management**

   (a) **[10 points]** Suppose we have an access matrix as shown below, where 'o', 'r', 'w' and 'e' mean "owner", "read only", "write only" and "execution only", respectively:

	File 1	File 2	File 3
User A	orwe	rw	rw
User B	r	orwe	e
User C	re	rw	orwe

   From this access matrix, construct a *access control list* and a *capability list.* **Make sure you will indicate clearly which answer is for the access control list and which is for the capability list. Otherwise, you will receive no credit.**

   (b) **[8 points]** Suppose a Unix inode has 10 direct disk block pointers, one single indirect block pointer, one double indirect block pointer, and one triple indirect block pointer. For convenience, each index table has $256 = 2^8$ entries and each block has 1K= 2^{10} bytes. What is the maximum size of a Unix file? **You must provide a complete calculation and your answer. Otherwise, you will receive no credit.**

   (c) **[10 points]** Consider a typical disk with an advertised average seek time 10 ms, rotation speed 7,200 rpm (*i.e.*, rotations per minute), 512-byte sectors, and 320 sectors per track. How much time does it take to read a complete sector? **You should provide your detailed calculation. Only showing a formula and/or an answer receives no credit.**

(d) **[16 points]** Suppose a disk has 30 cylinders numbered 0 (outer-most) to 29 (inner-most). In a sequence of I/O requests, the cylinders, in the order received, are 10, 6, 15, 18 and 9. The disk head is currently on cylinder 10 and moving direction is *inward*. Use the following disk head scheduling algorithms to determine the cylinders visited in correct order and the total number of cylinders crossed.

	Cylinders Visited					
Algorithm	1	2	3	4	5	*Cylinders Crossed*
FCFS	10					
SSTF	10					
SCAN	10					
LOOK	10					

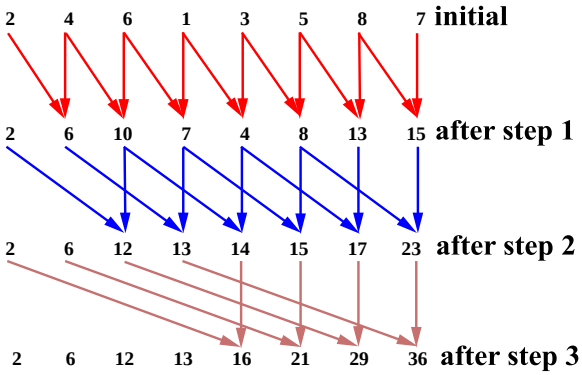
(e) **[12 points]** A file currently has 5 blocks, from block 0 to block 4. Assume that its FCB and directory are already in memory. There are a few more assumptions as shown blow:

- In the contiguous allocation case, assume that there is no room to grow in the beginning, but room to grow in the end, and the directory entry has a start pointer and a length value.
- In the linked allocation, a directory entry has a start pointer and an end pointer.
- In the FAT allocation, the FAT table is in memory and a directory entry has a start pointer to the FAT table.
- In the case of the indexed allocation, the index block is in memory and is large enough for further expansion.

How many disk I/O operations (*i.e.*, reads and writes) are required for contiguous, linked, FAT, and indexed (single-level) allocation strategies, if a new block is to be inserted between block 3 and block 4. Assume that the block information to be added is already in memory and that the directory and index will **not** be written back to disk after this insertion is done. **Note that there are four questions in this problem. Note also that you have to provide a complete and detailed elaboration of your calculation. Only providing an answer or vague elaboration will receive no credit.**

5. **Problem Solving:**

(a) **[30 points]** Given an array $x[\,]$ of 2^n elements $x[0]$, $x[1]$, ..., $x[2^n-1]$, the k-th prefix of x is $x[0]+x[1]+\cdots+x[k]$. We may use 2^n threads to compute *all* prefix values in n steps. If one thread is used, we need 2^n-1 steps. The following diagram shows a possible computation scheme for $2^3=8$ elements. Each step consists of taking two elements to compute a result. The first step computes the sums of adjacent elements (*i.e.*, gap $=1=2^0$). The second step computes the sums of the elements that are two elements away (*i.e.*, gap $=2=2^1$). The third step computes the sums of the elements that are 4 elements away (*i.e.*, gap $=4=2^2$). After 3 steps, we stop and all prefix values are computed. In general, if we have 2^n elements, step i computes the sums of elements that are 2^{i-1} elements away. After n steps, all prefix values are computed.



Assume that when thread T_i $(0 \le i \le 2^n-1)$ is created, it has the values of $x[i]$ and n in local variables s and n. The system has a mailbox-like **synchronous** channel $\mathcal{M}$ which is created before T_i's are created. Channel $\mathcal{M}$ has two primitives:

- `SEND(`$\mathcal{M}$`,` *receiver*`, int value)`: Here, *receiver* is the i in T_i. This function sends an integer stored in `value` to thread $T_{receiver}$ via channel $\mathcal{M}$.
- `RECEIVE(`$\mathcal{M}$`,` *receiver*`, int *value)`: Thread $T_{receiver}$ receives an integer value into variable `value` from channel $\mathcal{M}$.

With channel $\mathcal{M}$ and primitives `SEND()` and `RECEIVE()`, thread T_i executes n steps to compute the i-th prefix. In each step, T_i has an execution template of three sub-steps: **(1)** receives a value from channel $\mathcal{M}$, **(2)** adds the received value to local variable s, and **(3)** passes the result to an appropriate thread via channel $\mathcal{M}$. At the end of execution, local variable s has the i-th prefix. However, not all threads require all three sub-steps. For example, the last thread T_{2^n-1} does not send its result, and the first thread T_0 does not do any calculation.

To ensure threads are synchronized (*i.e.*, all threads being in the same step i), the system has a mechanism `sync()`. After finishing step i, every thread executes `sync()` to block itself until all other threads finish step i. Then, all threads are released so that they can start step $i+1$.

Use channel $\mathcal{M}$ and `SEND()` and `RECEIVE()` to write a prefix computation function. Recall that channel $\mathcal{M}$ has been created elsewhere. When thread T_i is created, it receives three items: **(1)** its ID i, **(2)** the n in 2^n-1 (*i.e.*, the total number of elements), and **(3)** the value of $x[i]$ stored in a local variable s of T_i. At the end of its execution, a thread prints out the following message:

```
Thread xx has prefix yy
```

where `xx` is the thread ID i and `yy` is the i-th prefix. **Array $x[\,]$ is not available to any thread.** You only need to write the prefix computation function and syntax is relatively unimportant. **You will risk lower grade if you do not supply an elaboration of your program logic and/or if your program can only work on a particular n (*e.g.*, $n=3$). Note that you can only use the channel primitives to implement semaphore primitives. You will receive no credit if you use any primitives other than channels and if your implementation has busy waiting, deadlock, and race conditions.**

Use this blank page for your answer

Grade Report

Problem		*Possible*	*You Received*
1	a	10	
	b	10	
2	a	8	
	b	10	
3	a	10	
	b	10	
	c	10	
	d	10	
	e	10	
	f	16	
	g	10	
4	a	10	
	b	8	
	c	10	
	d	16	
	e	12	
5	a	30	
Total		200	