CIS 520 - Final Exam – Fall 2012 Name: _____

Due Date: Thursday, December 13, 2012, 1:40 pm (N219D).

Multiple Choice:

1. (15 pts.) Consider a simple one-level paging system with the following parameters: each page is of size 1024 bytes, in a 16-bit logical address space, and each page table entry takes up 4 bytes in the page table. Each process can address up to 2^{16} bytes. Hint: $1024 = 2^{10}$.

   In each logical address, how many bits are used to represent the **page number**?
   a. 2
   b. 4
   c. 6
   d. 8
   e. 10

   In each logical or physical address, how many bits are used for the **offset** field?
   a. 2
   b. 4
   c. 6
   d. 8
   e. 10

   For each process, how many **logical pages** are there (expressed as a power of 2)?
   a. 2^6
   b. 2^8
   c. 2^{10}
   d. 2^{12}
   e. 2^{16}

   In this system, how large is the **page table** in bytes?
   a. 2^6
   b. 2^8
   c. 2^{10}
   d. 2^{12}
   e. 2^{16}

   With the following page table in which all entries are valid (pages are currently mapped into memory), the logical address 0x0C3F would be mapped to which physical address:
   a. 0x0A3F
   b. 0x283F
   c. 0x2A3F
   d. 0x0B3F
   e. none of the above

Page Number	Page Frame
0x00	0x0B
0x01	0x10
0x02	0x11
0x03	0x0A
..	..
0x3F	0x0C

2. (15 pts.) Consider the following three threads executing different paths of execution. Determine if and where deadlock can occur for each thread and what final values are possible. Assume that global variables d, e, and f are initialized to 1, and all locks are initially free. Circle all correct answer(s).

Line	Thread 1	Thread 2	Thread 3
1:	lock(mutex_d)	lock(mutex_e)	lock(mutex_d)
2:	lock(mutex_e)	lock(mutex_d)	
3:	e = d * 2	f = f + e	d = d * f
4:	unlock(mutex_e)	unlock(mutex_d)	
5:	unlock(mutex_d)	unlock(mutex_e)	unlock(mutex_d)

a. Deadlock can occur at Thread 1, Line 1; Thread 2, Line 1; Thread 3, Line 1.
b. Deadlock can occur at Thread 1, Line 2; Thread 2, Line 2; Thread 3, Line 1.
c. Deadlock can occur at Thread 1, Line 1; Thread 2, Line 2; Thread 3, Line 3.
d. Deadlock can occur at Thread 1, Line 3; Thread 2, Line 3; Thread 3, Line 1.
e. Deadlock can occur at Thread 1, Line 4; Thread 2, Line 4; Thread 3, Line 4.
f. Possible final values are e=2, f=3, and d=3.
g. The final values are always d = e = f = 1 because the system always deadlocks.
h. Possible final values are e=4, f=2, and d=2.
i. Possible final values are e=2, f=2, and d=2.
j. Deadlock cannot occur.

3. (15 pts.) Using RSA, a public key (e, n) and a private key (d, n) can be constructed from prime numbers, p = 11 and q = 17, by setting n = p*q = 187. If we select e = 7, compute the corresponding decryption key; e.g., d = _____.

Show how the data m = 22 would be encrypted using the encryption key (7, 187), and compute the ciphertext.

Show how the ciphertext c = 44 would be translated back into plaintext m, and how the plaintext value m can be computed efficiently by hand.

Short Answer:

4. (15 pts.) Assume you have a page reference string for a process running on a system with **m** physical pages available to the process. All of the process's pages are initially non-resident in memory. The page reference string has length **p** with **n** distinct page numbers occurring in it. The process is running uninterrupted.

   a. For the FIFO page replacement algorithm, suppose that the reference string is given by 1, 2, 3, … , **n**, 1, 2, 3, … , **n**, … (1 through **n** repeating to length **p**). If **m** = **n**, and **p** > 2***n**, how many page faults will occur? Explain briefly.

   b. For the same problem, but with **m** = **n-1**, how many page faults will occur?

   c. For any reference string with **p** > **n** and any page replacement algorithm, what is a lower bound on the number of page faults if none of the pages are currently in memory? Explain briefly.

   d. For any reference string with **p** > **n** and any page replacement algorithm, what is an upper bound on the number of page faults in none of the pages are currently in memory? Explain briefly.

   e. Give an example to show that the FIFO algorithm can suffer from Belady's anomaly.

5. (15 pts.) Process Model.

   a. Including the initial parent process, how many processes are created when the following program is executed? Draw the Process Model showing the parent-child hierarchy.

   b. What will be printed?

```
#include <stdio.h>
#include <unistd.h>

int main()
{
    int i, retval;

    for (i=1; i<=5; i++)
    {
        retval = fork();  // fork off a child process
        if (retval == 0)
        {
          printf("In child %d\n", i);
          return 0;
        }
    }
    printf("In parent\n");
    pause();  // wait for any signal to be received
}
```

   c. How many processes would be created if the "`return 0;`" statement was removed?

6. Deadlocks and Deadlock Avoidance:

   a. A system has **n** identical resources. Each process needs a maximum of 2 resources. What is the maximum number of processes that can exist in the system and still ensure that the system will not deadlock if all resource requests are granted immediately upon request?

   b. The Banker's Algorithm can be used to avoid deadlocks by granting resource requests only if the system remains in a safe state. Could a system with 5 identical resources and 4 processes, each needing a maximum of 2 resources each, ever reach an unsafe state, even without the Banker's Algorithm? Explain briefly. What about a system with 5 identical resources and 5 processes?

   c. Consider a system with four processes and three resource types A, B, and C, with instances of 6, 5, and 5, respectively. The current state of the system is as follows:

Max:	**A B C**	**Avail:**	**A B C**
	4 2 5		2 1 2
	2 4 4		
	1 2 1		
	6 1 0		
Alloc:	**A B C**	**Total:**	**A B C**
	1 1 1		6 5 5
	0 1 2		
	1 2 0		
	2 0 0		

   Is the system in a safe state? Explain briefly.

   d. Consider the system shown above in part c, if a request from the second process (currently holding 1 1 0) for 2 instances of resource type **A** is granted, will the system still be in a safe state? Is the system deadlocked? Explain briefly.

   e. Again, starting from the configuration shown in part c, , if a request from the last process (currently holding 2 0 0) for 2 instances of resource type **A** is granted, will the system still be in a safe state? Is the system deadlocked? Explain briefly.

7. (10 pts.) Security Questions:

   a. What is the difference between authentication and authorization?

   b. Why is authorization useless if we don't have authentication?

   c. If the following code is used to check user passwords, an attacker can determine if each letter is correct by timing how long the password checking routine executes before reporting failure:

```
for (i=0; i<16; i++)
     if (enteredPassword[i]!= userPassword[i])
          return(error);
```

   Rewrite the code to eliminate this vulnerability below: