# Homework 7 for CSC246

Homework 7 appropriately consists of 7 questions and is your last homework. Respond to each question and submit your work electronically using the link on the course web page. Submit your answers in either plain text, Microsoft Word documents (doc or docx is fine), pdf, or rtf.

If you need help, please contact the TA at once and arrange to get help. The TA's email address is on the syllabus.  
  
This homework is due on the date mentioned on the main website **by midnight.** This homework contains 100 points and is 10% of your total course grade.

*15 points*

## Question 1: Deadlocks

Consider a system consisting of four resources of the same type that are shared by three processes, each of which needs at most two resources. Show that the system is deadlock free. *Hint: Demonstrate that the system is always in a safe state*.

If three processes each need maximum of two resources and the system has four of those resources total, this means one of the processes already have the amount of resources it needs to complete. After the process completes, more resources are freed, and the other process can complete. Since the system is only deadlocked if a process cannot access the maximum amount of resources it needs, the system is considered as deadlock free.

*20 points*

## Question 2: Banker's Algorithm

Consider the following snapshot of a system.

**Current**

**Allocation Max Available**

**A B C A B C A B C**

**P0 0 1 0 0 2 0 1 1 1**

**P1 2 0 0 2 1 2**

**P2 3 0 3 3 0 4**

**P3 2 1 1 3 1 1**

**P4 0 0 2 0 0 4**

Answer the following questions using the banker's algorithm.

What is the content of the *Need* matrix?

Need [ i, j ] = Max[ i, j ] – Allocation [ i, j ]

ABC

P0 0 1 0

P1 0 1 2

P2 0 0 1

P3 1 0 0

P4 0 0 2

Is the system in a safe state? Demonstrate why or why not.

If the available resource is allocated to the processes with the order P3 > P2 > P4 > P0 > P1, then every process will get what it needs.

If a request from process P1 arrives for (0,1,0) can the request be granted immediately? Why or why not?

Yes, because the available resources are (1,1,1).

*15 points*

## Question 3: Deadlocks

List the four necessary conditions for a deadlock.

1. Mutual exclusion
2. No preemption
3. Hold and wait
4. Circular wait

Explain how all four are met when 5 Dining Philosophers simultaneously pick up the chopstick on their left.

Mutual Exclusion – Since there are only 5 chopsticks and only 2 philosophers can eat at the same time, but none of them are allowed to finish eating so the philosophers cannot use the resources properly.

No preemption – None of the philosophers will put down their chopsticks until they finish eating

Hold and wait – They are holding one of the two chopsticks and waiting on someone to give up the other chopstick.

Circular Wait – They are all waiting on the philosopher beside them to give up their chopstick.

If an extra chopstick is provided, deadlock will not occur. Explain why (Hint: what condition is altered?)

Hold and wait will be altered. If an extra chopstick is provided, one philosopher will have two chopsticks and will no longer be waiting. When he finishes eating he can give his chopsticks to other philosophers to repeat the cycle.

Explain how resource ordering could be used to prevent deadlock with the 5 philosophers (and 5 chopsticks).

Assume the philosophers are sitting around a table in order from 0 – 4. Each philosopher can request the chopstick on his left since their number is less than the one on the right but the philosopher 4 will grab the one on the right first since the number on right is rank lower.

*15 points*

## Question 4: More Deadlocks

Consider the following pseudo code for two threads using mutex locks. Assume the mutex locks are counting semaphores initialized to a count of 1.   
 **Thread 1                  Thread 2**  
wait (mutex1)           wait (mutex2)  
wait (mutex2)           wait (mutex1)  
...                              ...  
signal (mutex2)         signal (mutex1)  
signal (mutex1)         signal (mutex2)  
  
Explain a scenario leading to deadlock in the above code. Identify the 4 conditions.

Thread 1 will obtain mutex 1 and thread 2 will obtain mutex 2. Then thread 1 will wait on mutex 2 and thread 2 will wait on mutex 1(hold and wait), but these mutex will be held by the other one (circular wait). So, both threads cannot continue until they have both (mutual exclusion) and there will be no signal to other one (no preemption).

Explain how the code can be modified using resource ordering to insure a deadlock is prevented.

The code can be modified by changing the order in thread 2. The process will have mutex 1 before they get mutex 2. Then only one process will be working at a time since one will have mutex 1 and the other will be waiting on mutex 1.

*10 points*

## Question 5 : Paging

Consider a logical address space of 16 pages of 512 bytes each, mapped onto a physical memory of 64 frames.

How many **bits** are in the logical address?

16 = 24

512 = 29

4+9 = 13

13 bits are in the logical address

How many **bits** are in the physical address?

64 = 26

6+9 = 15

15 bits are in the physical address

On what **page** would address 1025 reside?

Page 2

How large is a frame on this system?

512 bytes

*15 points*

## Question 6: Paging

Consider the page reference string below. If 3 frames are allocated for the process generating these references, a page will have to be replaced when page 4 is needed. Provide the page number that would be replaced for the specific algorithms.

1 2 1 2 3 4 3 1 2 3

 FIFO: \_\_1\_\_\_\_\_\_

 LRU: \_\_\_1\_\_\_\_\_

 Optimal: \_\_2\_\_\_\_\_

*10 points*

## Question 7: Paging

Consider the following address sequence (decimal values):

0100,0432,0101,0612,0102,0504,0101,0611,0803,0104,0101,0610,  
0702,0103,1004,0101,0609,0102,0105

At 200 (decimal) bytes per page, provide the corresponding page reference string. The page reference string is a list of the pages that are referenced.

1, 3, 1, 4, 1, 3, 1, 4, 5, 1, 1, 4, 4, 1, 6, 1, 4, 1, 1