Learning Goals for course Concurrent Programming

Main theme	***Prerequisites***	***Approaching Learning Goals***	***Reaches Learning Goals***	***Deepens Learning Goals***
Concurrency and problems caused by concurrency. *Can explain where concurrency appears in programs and systems, how the execution of concurrent programs is described and what problems may result from concurrency.*	Can explain processor operation when it executes machine instructions. (Computer Organization I) Can explain how two programs may advance concurrently at machine language level. Can explain how concurrent advancement is based on device interrupts and operating systems scheduling decisions. (Computer Organization I)	Can explain shared memory use of multiple threads/processes. Can explain the need for concurrency in applications. Can explain what types of actions result in concurrent actions in HW and SW level. Can give concurrency examples at different system levels. Can explain the meaning of different execution scenarios when studying the execution of concurrent programs. Can give examples of general problems that may result from erroneous concurrent solution.	Can explain the indeterminate nature of concurrent processes and its effects. Can explain the goals of concurrency (e.g., advantages, disadvantages, different levels of concurrency, and proving concurrent programs correct or faulty). Can explain the requirements for concurrent solutions (e.g., definition of correctness and correctness in all scenarios). Can take into consideration the effect of systems architecture to concurrency.	Can explain the different nature of concurrency in HW, SW and network level operations. Can estimate how well an application may use concurrency. Can maximize concurrency in application.

Main theme	***Prerequisites***	***Approaching Learning Goals***	***Reaches Learning Goals***	***Deepens Learning Goals***
Fundamental concepts and models in concurrency. *Can explain the problem areas, the general solution models and the commonly used model examples for concurrent problems.*	Can program simple one-threaded programs. (Introduction to Programming) Can explain how OS works when scheduling processes and managing resources. (Computer Organization I)	Can describe with examples • fundamental concepts in concurrent execution (e.g., synchronic, asynchronic, atomic, critical section, synchronization, communication, deadlocking, lock variable, busy wait and suspended wait), • fundamental concurrent programming models (e.g., mutual exclusion problem, readers-writers, producer-consumer, client-server, boom synchronization, user class based synchronization, active server), and • classical examples in concurrent programming (e.g., dining philosophers, sleeping barber, bakery algorithm). Can explain semaphore and monitor structures, and use them properly in applications. Can explain the meaning of mutual exclusion, synchronization and communication problems in concurrent programming. Can design a mutual exclusion problem solution that is best suitable for current environment and application. Can explain with examples how deadlocks are created as well as explain the necessary and sufficient conditions for them. Can explain at algorithm level how deadlocks are found and recovered from.	Can explain how program behaviour causes concurrency problems. Can apply fundamental models to new problems. Can explain special features of semaphores and monitors and use them properly in applications. Can program concurrency control at application, programming language and operating system level. Can select correct methods for mutual exclusion, synchronization and communication problem solutions. Can reason with invariants the correctness of algorithms applying fundamental models. Can reason the limited possibilities in handling deadlocks in real applications. Can explain how deadlocks can be prevented.	Can evaluate the usability of fundamental models in various HW and SW platforms. Can program semaphores and monitors with lock variables. Can program monitors with semaphores. Can prove correctness of complex concurrent programs.

Main theme	***Prerequisites***	***Approaching Learning Goals***	***Reaches Learning Goals***	***Deepens Learning Goals***
Concurrent programming in distributed systems. *Can explain the special features of distributed systems for concurrent programming and how fundamental models can be implemented in distributed system.*	Can explain how operating system transmits messages from one process to another. (Computer Organization I) Can explain how subroutines are implemented. (Computer Organization I)	Can explain concurrent programming fundamental concepts in distributed systems (e.g., message passing, channels, RPC, guarded commands, client-servers). Can explain how message passing through channels works and apply message passing into fundamental model solutions in distributed systems. Can explain how remote procedure calls, guarded commands and client-server systems work.	Can algorithmically implement fundamental models in distributed systems based on peer communication. Can explain the similarities and differences between monitor and message based servers. Can explain how remote procedure call and guarded commands work. Can explain how normal and rendezvous based client server systems work. Can algorithmically implement guarded commands both with threads and with guarded statements.	Can explain how inter-process message communication can be generalized to network level. Can explain how CSP works and what it is used for.
Concurrent programming in practice. *Can write small concurrent programs with Java.*	Can write small one-threaded Java programs. (Introduction to Programming)	Can write multi-threaded Java programs using shared memory with mutual exclusion. Can describe the most common error types of Java concurrent programming.	Can write small Java programs utilizing concurrency control. Can determine whether simple concurrent programs are correct or not.	Can write distributed concurrent Java programs. Can test concurrent programs and prove them correct.