Operating systems
Question Bank (Unit 2)

Q1. Explain why implementing synchronization primitives by disabling interrupts is not appropriate in a single processor system if the synchronization primitives are to be used in user-level programs.

Q2. Explain why interrupts are not appropriate for implementing synchronization primitives in multiprocessor systems.

Q3. *Write two short methods that implement the simple semaphore wait() and signal() operations on global variable S.*

Q4. Explain the difference between the first readers–writers problem and the second readers–-writers problem.

Q5. Describe the dining-philosophers problem and how it relates to operating systems.

Q6. 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.

Q7 Consider a system consisting of m resources of the same type being shared by n processes. A process can request or release only one resource at a time. Show that the system is deadlock free if the following two conditions hold (a)The maximum need of each process is between 1 and m resources (b) The sum of all maximum needs is less than m + n

Q8 Consider the version of the dining-philosophers problem in which the chopsticks are placed at the centre of the table and any two of them can be used by a philosopher. Assume that requests for chopsticks are made one at a time. Describe a simple rule for determining whether a particular request can be satisfied without causing deadlock given the current allocation of chopsticks to philosophers.

Q9 Consider again the version of the dining-philosophers problem in which the chopsticks are placed at the centre of the table . However, assume that requests for chopsticks are made one at a time. Assume now that each philosopher requires three chopsticks to eat. Resource requests are still issued one at a time. Describe some simple rules for determining whether a particular request can be satisfied without causing deadlock given the current allocation of chopsticks to philosophers.

Q10. Describe the four conditions that must hold simultaneously in a system if a deadlock is to occur.

Q11. Is it possible to have concurrency but not parallelism? Explain.

Q12. A system with two dual-core processors has four processors available for scheduling. A CPU-intensive application is running on this system. All input is performed at program start-up, when a single file must be opened. Similarly, all output is performed just before the program terminates, when the program results must be written to a single file. Between start-up and termination, the program is entirely CPU-bound. Your task is to improve the performance of this application by multithreading it. The application runs on a system that uses the one-to-one threading model (each user thread maps to a kernel thread).

- *How many threads will you create to perform the input and output? Explain.*
- *How many threads will you create for the CPU-intensive portion of the application? Explain.*

Q13 Consider a multiprocessor system and a multithreaded program written using the many-to-many threading model. Let the number of user-level threads in the program be more than the number of processors in the system. Discuss the performance implications of the following scenarios.

a. *The number of kernel threads allocated to the program is less than the number of processors.*

b. *The number of kernel threads allocated to the program is equal to the number of processors.*
c. *The number of kernel threads allocated to the program is greater than the number of processors but less than the number of user-level threads.*

*Q14 Pthreads provides an API for managing thread cancellation. The pthread_setcancelstate() function is used to set the cancellation state. Its prototype appears as follows: pthread_setcancelstate(int state, int *oldstate). The two possible values for the state are PTHREAD_CANCEL_ENABLE and PTHREAD_CANCEL_DISABLE. Provide examples of two operations that would be suitable to perform between the calls to disable and enable thread cancellation.*

Q15. Provide two programming examples in which multithreading does not provide better performance than a single-threaded Solution.

Q16.Under what circumstances does a multithreaded solution using multiple kernel threads provide better performance than a single-threaded solution on a single-processor system?

Q17 Which of the following components of program state are shared across threads in a multithreaded process?

a. *Register values*
b. *Heap memory*
c. *Global variables*
d. *Stack memory*

Q18. Using the Windows scheduling algorithm, determine the numeric priority of each of the following threads
a. A thread in the REALTIME_PRIORITY_CLASS with a relative priority of HIGHEST.
b. A thread in the NORMAL_PRIORITY_CLASS with a relative priority of NORMAL.

c. thread in the HIGH_PRIORITY_CLASS with a relative priority of ABOVE_NORMAL.

Q19. Assuming that no threads belong to the REALTIME_PRIORITY_CLASS and that none may be assigned a TIME_CRITICAL priority, what combination of priority class and priority corresponds to the

highest possible relative priority in Windows scheduling?

Q20. Assuming that no threads belong to the REALTIME_PRIORITY_CLASS and that none may be assigned a TIME_CRITICAL priority, what combination of priority class and priority corresponds to the highest possible relative priority in Windows scheduling?