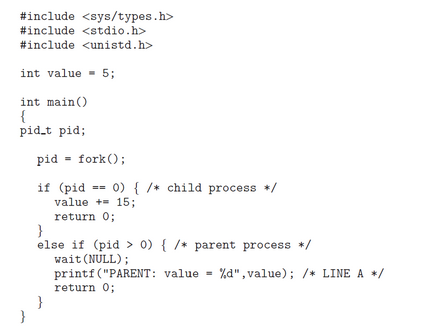
Due: Sun Oct 25, 2020 11:59pm

**Instructions**

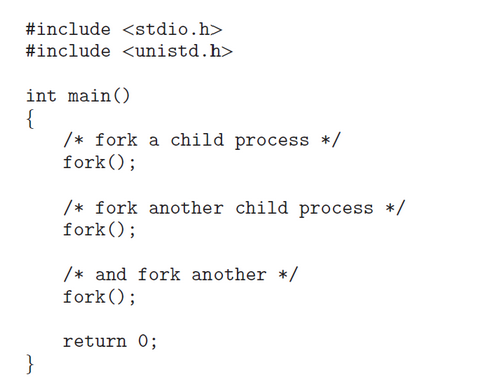
Provide concise answers to the following questions. Submission must be a Word or PDF document and must contain the question text above your answer.

**Chapter 3 Questions**

1. Using the program shown below, explain what the output will be at LINE A.



2. Including the initial parent process, how many processes are created by the program shown below?



3. Original versions of Apple’s mobile iOS operating system provided no means of concurrent processing. Discuss three major complications that concurrent processing adds to an operating system.

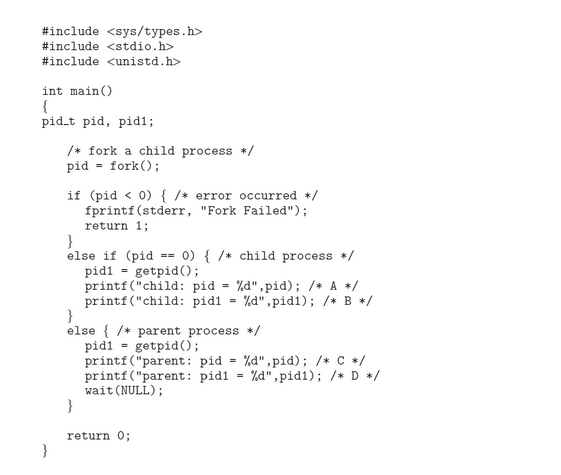
4. When a process creates a new process using the fork() operation, which of the following states is shared between the parent process and the child process?

     a. Stack  
     b. Heap  
     c. Shared memory segments

5. Describe the actions taken by a kernel to context-switch between processes.

6.  Explain the role of the init (or systemd) process on UNIX and Linux systems in regard to process termination.

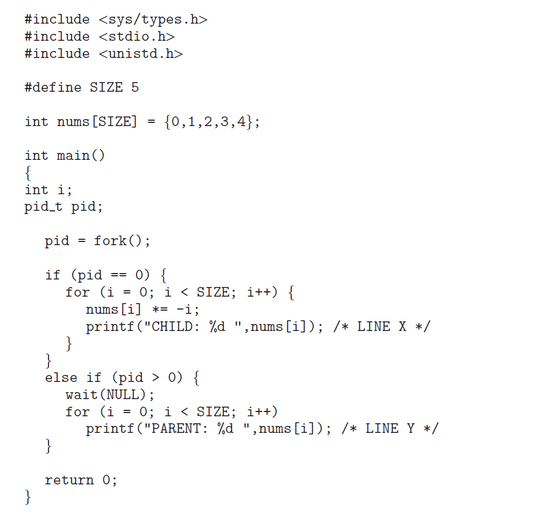
7. Including the initial parent process, how many processes are created by the program shown below?



8. In the program above, explain the circumastances under which the line of code marked printf("Line J") will be reached.

9. What are the benefits and the disadvantages of each of the following? Consider both the system level and the programmer level.  
     a. Synchronous and asynchronous communication   
     b. Automatic and explicit buffering  
     c. Send by copy and send by reference   
     d. Fixed-sized and variable-sized messages

10. What will be the output at lines X and Y in the program below?



**Chapter 4 Questions**

11.  Provide three programming examples in which multithreading provides better performance than a single-threaded solution.

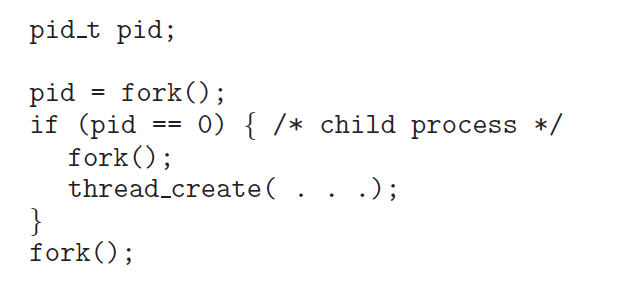
12.  Using Amdahl’s Law, calculate the speedup gain of an application that has a 60 percent parallel component for (a) two processing cores and (b) four processing cores.

13. What are two differences between user-level threads and kernel-level threads? Under what circumstances is one type better than the other?

14. Describe the actions taken by a kernel to context-switch between kernel-level threads.15. What resources are used when a thread is created? How do they differ from those used when a process is created?

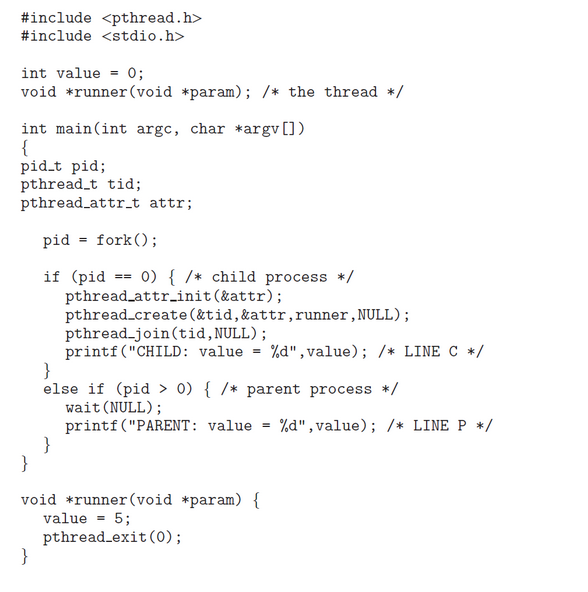
15. In Chapter 3, we discussed Google’s Chrome browser and its practice of opening each new tab in a separate process. Would the same benefits have been achieved if, instead, Chrome had been designed to open each new tab in a separate thread? Explain.

16. Consider the following code segment:



     a.  How many unique processes are created?  
     b.  How many unique threads are created?

17. The program shown below uses the Pthreads API. What would be the output from the program at LINE C and LINE P?

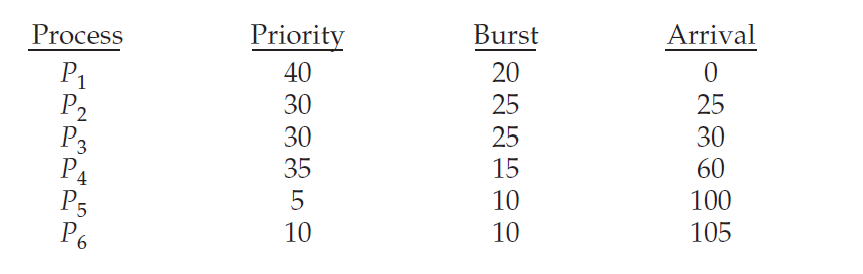


**Chapter 5 Questions**

18. A CPU-scheduling algorithm determines an order for the execution of its scheduled processes. Given n processes to be scheduled on one processor, how many different schedules are possible? Give a formula in terms of n.

19. Explain the difference between preemptive and non-preemptive scheduling.

20. The following processes are being scheduled using a preemptive, round robin scheduling algorithm.



Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. In addition to the processes listed below, the system also has an idle task (which consumes no CPU resources and is identified as Pidle). This task has priority 0 and is scheduled whenever the system has no other available processes to run. The length of a time quantum is 10 units. If a process is preempted by a higher-priority  
process, the preempted process is placed at the end of the queue.  
     a. Show the scheduling order of the processes using a Gantt chart.  
     b. What is the turnaround time for each process?  
     c. What is the waiting time for each process?  
     d. What is the CPU utilization rate?

21. What advantage is there in having different time-quantum sizes at different levels of a multilevel queueing system?

22. Many CPU-scheduling algorithms are parameterized. For example, the RR algorithm requires a parameter to indicate the time slice. Multilevel feedback queues require parameters to define the number of queues, the scheduling algorithms for each queue, the criteria used to move processes between queues, and so on. These algorithms are thus really sets of algorithms (for example, the set of RR algorithms for all time slices, and so on). One set of algorithms may include another (for example, the FCFS algorithm is the RR algorithm with an infinite time quantum). What (if any) relation holds between the following pairs of algorithm sets?  
     a. Priority and SJF  
     b. Multilevel feedback queues and FCFS  
     c. Priority and FCFS  
    d. RR and SJF

23. Suppose that a CPU scheduling algorithm favors those processes that have used the least processor time in the recent past. Why will this algorithm favor I/O-bound programs and yet not permanently starve CPU-bound programs?

24. The traditional UNIX scheduler enforces an inverse relationship between priority numbers and priorities: the higher the number, the lower the priority. The scheduler recalculates process priorities once per second using the following function:

Priority = (recent CPU usage / 2) + base

where base = 60 and recent CPU usage refers to a value indicating how often a process has used the CPU since priorities were last recalculated.

Assume that recent CPU usage for process P1 is 40, for process P2 is 18, and for process P3 is 10. What will be the new priorities for these three processes when priorities are recalculated? Based on this information, does the traditional UNIX scheduler raise or lower the relative priority of a CPU-bound process?

25. 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 NORMAL

     b. A thread in the ABOVE NORMAL PRIORITY CLASS with a relative priority of HIGHEST  
     c. A thread in the BELOW NORMAL PRIORITY CLASS with a relative priority of ABOVE NORMAL