Sample Questions for Quiz 2

1. *Threads* are sometimes also called
   a. heavyweight processes.
   b. lightweight processes.
   c. process controls.

2. Each application that is currently active in a system will have
   a. at least two related processes, one for input and one for output.
   b. had at least one process to get it started, but the process may have terminated when a thread took over.
   c. at least one related process.

3. Consider the number of processes and processors in a typical system. The likely relationship between these is
   a. that there are more processes than processors.
   b. impossible to predict.
   c. that there are always exactly the same number of processes as there are processors.

4. Using multiple processes in the implementation of a single application may be appropriate to take advantage of multiple processing elements in a computer system. What other reason might motivate the use of multiple processes in a single application?
   a. Some of the system’s processors would be underutilized if we did not have a sufficiently large number of processes.
   b. It is often easier to develop an algorithm for a problem if each of several tasks can be assigned to separate processes.
   c. In systems that maintain resource accounting records, using the same amount of processor time to complete some work, but dividing it among multiple processors, will be less expensive than doing all the work on a single processor.

5. A disadvantage of using multiple processes to solve a problem is that
   a. it will likely require more energy to run the program that uses multiple processes.
   b. there is additional overhead needed in the operating system to switch the processor among the various processes.
   c. it will likely require more time to write the program that uses multiple processes.

6. Assume three processes are executed sequentially, one after another, on a system with specific sets of data. Then assume these three processes are executed at the same time with exactly the same data. Which approach to execution will require less total time?
   a. Execution of the three processes at the same time will take longer if the processes each read data from the same disk.
   b. Since they are executing exactly the same instructions and processing the same data in each case, the total time will be exactly the same regardless of how the processes are executed.
   c. It is impossible to tell.

7. A *program* is
   a. one or more processes being executed by one or more processors.
   b. the set of instructions and the contents of the processor registers while the processor is executing the set of instructions.
   c. a static set of instructions and data for a processor.
8. A *process* is
   a. one or more programs being executed by one or more processors.
   b. the activity of a processor in carrying out the actions associated with a set of instructions.
   c. the data and the contents of the processor registers while the processor is executing a program.

9. Which of the following is *not* part of a process?
   a. the set of privileges associated with the user on whose behalf a process is being executed.
   b. the quotas (if any) associated with the use of various system resources by the process.
   c. the processor on which the process is being executed.

10. Processes may be created in UNIX systems by using which of the following system calls?
    a. pipe
    b. fork
    c. execvp

11. Consider the hierarchy of processes in a UNIX system. What data structure best characterizes this hierarchy?
    a. a general tree
    b. a doubly-linked list
    c. a singly-linked list

12. In what position in the hierarchy of processes in a UNIX system will we find processes that have no child processes of their own?
    a. They will be root nodes in the forest of trees.
    b. They will be leaf nodes.
    c. They will be interior nodes.

13. Why must some components of an operating system usually be written in assembly (or assembler) language?
    a. Assembly language code is the only code that can be executed directly by a processor at boot time.
    b. High-level languages are generally designed to be independent of specific processor architecture, and thus cannot be used to write code to control the unique features of a given processor.
    c. The code generated by compilers for high-level language is just too inefficient, in general.

14. Recall that special trap instructions (like INT on Intel X86 processors and SVC on IBM mainframes) are often used to invoke system calls. For high-level languages, these instructions are
    a. generated by the compiler.
    b. only executed by a special process charged with the interface between user processes and the kernel of the operating system.
    c. usually found in libraries of functions specific to the language and the operating system being used.

15. How is possible that the *fork* system call can be used to create processes on so many different UNIX systems when we know that different machine language instructions are used by different processor architectures to invoke system calls?
    a. Every processor ever built can interpret a limited subset of instructions that are used to allow the execution of a limited set of system calls, including *fork*.
    b. A *fork* function is provided for each unique system (and language), and it is this function that actually includes the specific machine language instruction(s) necessary to invoke the *fork* system call.
    c. The invocation of system calls is done from application programs using scripts that are interpreted by special code in the BIOS of every system.
16. The typical states in which any process exists are called
   a. ready, running, and terminated.
   b. ready, running, and blocked.
   c. running, not-running, and blocked.

17. A process in the running state
   a. may, or may not, have all the resources it needs to use a processor.
   b. is actively using a processor.
   c. is also in the ready state.

18. The number of processes in the running state
   a. is never greater than the number of processors on a system.
   b. is never equal to the number of processors on a system.
   c. is always less than the sum of the number of processes in the blocked and ready states.

19. A process in the ready state
   a. has all resources required for execution except a processor.
   b. has the use of a processor, but may lack other resources needed for execution.
   c. has all resources (including a processor) needed for execution, but may be waiting for one or more explicit processes to complete execution before it can be run.

20. Immediately before a process enters the running state
   a. it must have been waiting for user input.
   b. it must be in the ready state.
   c. it must have done a P or down operation on a semaphore.

21. A process in the blocked state
   a. is not capable of being executed because it does not possess one of the resources required for execution.
   b. is preventing another process from executing.
   c. can move directly to the running state when it obtains a needed resource.

22. Processes in the blocked state are sometimes said to be
   a. dead.
   b. terminated.
   c. sleeping.

23. If a process is waiting on keyboard input from a user, then it will be in the ____ state.
   a. blocked
   b. zombie
   c. terminated

24. Which of the following is not a valid state transition for a process?
   a. running to blocked
   b. blocked to running
   c. blocked to ready

25. When a processor becomes available to execute a process, the operating system’s scheduler selects an appropriate process from those in
   a. the state of being created.
   b. the ready state.
   c. the running state.
A process is moved from the running state to the blocked state
a. when it has completed execution.
b. when it requests a resource that is not currently available, but which will become available in the future.
c. when it has consumed too much processor time.

A process is moved from the blocked to the ready state
a. when the idle process terminates.
b. when the resource on which it has been waiting becomes available.
c. when all other processes with higher priorities are also blocked.

A process is moved from the running state to the ready state
a. when it is ready to move to the next stage of execution.
b. when the scheduler decides another process is more suitable for execution (for example, a ready process has higher priority than a running process).
c. when it has completed all input operations.

A request by a process for the current time of day
a. will not likely cause the process to block.
b. will cause a process to be moved to the ready state until the next 100-millisecond clock transition.
c. will not cause a process to block as long as at least one other process has made the same request within the last second.

When a process has consumed its fair share of processor time, the operating system may move it to the _____ state.
 a. sleep
 b. ready
 c. terminated

When a process is moved out of the running state
a. the processor’s registers are reset to the values they had when the process entered the running state.
b. state information (processor register contents, for example) is preserved so it can be restored when the process is next selected for execution.
c. the processor’s registers are set to default values in preparation for the execution of the next suitable process.

When a process is moved into the running state
a. all other processes are moved to the blocked state.
b. all resources on which blocked processes are waiting are made available to the process being moved to the running state.
c. the processor registers are set to the values they containing when the process was last moved out of the running state.

The act of moving one process out of the running state and replacing it with another is usually called
a. an event horizon.
b. a context switch.
c. a process refresh.

When a process is moved from the blocked state to the ready state,
a. no context switch is required.
b. the content of all processor registers is restored.
c. the content of all processor registers is saved.
35. Where is the unique identification of a process normally found?
   a. the process table entry for the process
   b. the system credentials register
   c. the entry for the process preceding it in the ready state

36. Interrupts are used to notify the processor of various events, usually those related to the hardware. For
    example, notifying the processor that a disk input/output operation has completed is generally sent as
    _____ notification.
   a. a preemptive
   b. an asynchronous
   c. a low-priority

37. The event being reported to the processor by an interrupt
   a. is always related to the instruction whose execution was just completed.
   b. is always related to the last process moved to the running state.
   c. may have nothing whatever to do with the process that is currently being executed.

38. The priority associated with an interrupt
   a. is higher for faster or more important devices than for slower or less important devices.
   b. is always one smaller than the priority associated with the process currently being executed.
   c. is always one larger than the priority associated with the process currently being executed.

39. If an interrupt is masked
   a. it will be discarded and never reported to the processor.
   b. it will be reported to the processor only after all other masked interrupts have been reported.
   c. it will not be reported to the processor, but it remains pending.

40. If interrupts are globally disabled, then
   a. no input/output operations can be completed.
   b. interrupts from most devices are prevented from being recognized by the processor.
   c. no processor can execute any input/output instructions.

41. An interrupt handler
   a. is the set of instructions executed as a result of an interrupt being reported to a processor.
   b. runs only when a process is blocked.
   c. is the user-mode code that responds to the interrupt resulting from a processor fault.

42. An interrupt handler is not allowed to
   a. execute input/output instructions.
   b. block.
   c. be preempted.

43. When an interrupt occurs, the type of interrupt is used to
   a. determine the location of the interrupt handler.
   b. determine which process will handle the interrupt.
   c. determine whether one or two additional instructions will be completed before the interrupt handler
       is executed.

44. An interrupt vector is
   a. a one-dimensional array containing the addresses of interrupt handlers, indexed by interrupt number,
      and used in many processor architectures.
   b. the directed path through the code or instructions comprising an interrupt handler.
   c. the action of the processor in selecting an interrupt handler to be executed.
45. A trap
   a. is a section of code used to identify and thus prevent the execution of various Trojan-horse viruses.
   b. is a type of interrupt that can be disabled by a user process.
   c. occurs synchronously with the execution of an instruction.

46. A trap can be generated for each of the following reasons except
   a. attempting to divide by zero.
   b. executing an INT instruction on an Intel x86 processor.
   c. completing a disk operation.

47. The process which generates a fault
   a. is the last process moved to the blocked state.
   b. is the currently executing process.
   c. is the process that was most recently terminated.

48. The operating system frequently translates a trap into
   a. a request to terminate the current process.
   b. a signal to the currently-executing process.
   c. a decrement of the number of such traps that were originally permitted when the currently-executing process was started.

49. When a trap or an interrupt is successfully recognized by the processor,
   a. the processor is always placed in supervisor mode.
   b. the priority of the current process is lowered so it is less than the priority of the trap or interrupt.
   c. the idle process is executed until the trap or interrupt has been processed.

50. A signal is
   a. always sent to the last process that was moved to the ready state.
   b. generated whenever any key is pressed on the keyboard.
   c. a software report to a process that an exceptional condition has occurred.

51. In most cases, sending a signal to a process that does not have an appropriate handler for that signal will
   a. cause the signal to be masked.
   b. cause termination of the process.
   c. cause the last signal handler registered for the process to handle the signal.

52. Which of the following will usually cause the generation of a SIGINT signal in a UNIX system?
   a. pressing any key on the keyboard
   b. exceeding the allowable processor time
   c. pressing the control-C key

53. Which of the following UNIX functions may be used to register a signal handler to deal with a floating point exception?
   a. signal
   b. exhandle
   c. register

54. Conceptually, each process has how many threads when it begins execution?
   a. two
   b. a variable number, depending on how the process was created
   c. one

55. Which of the following resources does a thread not share with other threads in the same process?
   a. its primary memory
   b. the resource usage quotas
   c. its stack
56. A thread represents  
   a. a separately scheduled unit of work for a processor.  
   b. one of several alternate process versions; the one that is executed depends on the particular processor  
      being used.  
   c. the connection between a signal and the signal handler that is registered to handle occurrences of that  
      signal.

57. One advantage of threads over processes is that  
   a. individual threads may have different owners, and hence may have different permissions with respect  
      to regions of memory.  
   b. multiple threads share the same address space.  
   c. multiple threads are scheduled as a single thread.

58. If threads are implemented as part of the kernel of an operating system, then  
   a. their scheduling can not be done as effectively as if they were implemented as part of a user library.  
   b. they will not be able to use trap-generating instructions to request operating system services (that is,  
      they will not be able to make system calls).  
   c. a context switch is required each time a processor is switched between threads, even if the threads  
      belong to the same process.

59. If threads are implemented through user library functions, then  
   a. providing the memory sharing required for threads will be much more difficult than if a kernel thread  
      implementation was used.  
   b. there is much less overhead (due to context switches) required when switching from one user thread  
      to another in the same process.  
   c. the kernel of the operating system will be able to more easily identify which threads are blocked and  
      which are not blocked.

60. The acronym IPC stands for _______.  
   a. interprocess communication  
   b. interrupt processing codes  
   c. interprocess consumption

61. A race condition exists when  
   a. two or more threads/processes are attempting to use a set of resources in a manner that causes each  
      of them to wait on a resource that one of the others holds.  
   b. two or more threads/processes attempt to use a resource in a manner that doesn’t guarantee consistency,  
   c. none of the above.

62. When we say an operation is atomic we mean  
   a. interrupts are disabled during its execution.  
   b. all parts of the operation are completed without the possibility of any other operation on any processor  
      manipulating the resources being used by the operation.  
   c. none of the above.

63. A sequence of instructions that modifies, updates or accesses a resource shared by multiple threads or  
    processes is called  
   a. a threaded section.  
   b. a critical process.  
   c. a critical section.
64. The mutual exclusion problem
   a. was first solved by Donald Knuth.
   b. requires special hardware for its solution.
   c. deals with techniques designed to guarantee that code segments accessing the same resources are executed atomically.

65. The most obvious condition necessary for a good solution to the race condition problem is that no two processes may be
   a. simultaneously inside their critical sections.
   b. executing at the same time (on a system with multiple processors).
   c. in the ready state at the same time.

66. What assumptions may we make about the speed with which processes are being executed by processors, and the number of processors?
   a. At most two processes may be executed at differing speeds; that is, if there are N processes, N-1 of them must be executed at the same speed.
   b. All processes must be executed at the same speed.
   c. No assumptions may be made.

67. Which of the following is not a necessary condition for a good solution to the mutual exclusion problem?
   a. No process should have to wait forever to enter its critical section once it has indicated a desire to do so.
   b. No assumptions may be made about the speeds of, or the number of processors.
   c. No process should have to limit the size of the memory region it is using.

68. The ability to disable recognition of interrupts (in particular, timer interrupts) can sometimes be used to implement a solution to the critical section problem. This solution will not work if
   a. the processes are all running in supervisor, or kernel mode.
   b. there are multiple processors.
   c. we are using threads instead of processes.

69. There are some proposed solutions to the mutual exclusion problem that utilize a lock variable. This variable is set to 0 if the shared resource is not being used, and set to 1 when the resource is in use. So if we want to use the resource, we have but to check the lock variable. If it is 1, we continually repeat the test until the lock variable becomes 0. Then we set it to 1, access the shared resource, and set the lock variable back to 0. What can go wrong with this solution?
   a. The solution, as presented, will not work if there are more than two processes.
   b. There is a race condition present. Multiple processes (or threads) could check the lock variable, find it 0, and then set it to 1 and enter their critical sections.
   c. Multiple processes cannot simultaneously check the value of the lock variable, meaning it is possible for the proper process to miss its opportunity to enter its critical section, resulting in no processes being able to continue.

70. Solutions to the mutual exclusion problem that require a process to check a lock variable repeatedly until it has a specified value, even if implemented correctly (that is, if they prevent multiple processes from entering their critical sections at the same time), still have a significant problem. What is that problem?
   a. The solution will not work correctly on systems with multiple processors.
   b. The process that is waiting to enter its critical section is continually consuming the processor resource, checking and checking again, significantly delaying other processes from using the processor and potentially relinquishing their use of the desired resource.
   c. The memory being accessed by the process checking the lock variable cannot be accessed by any other processes.
71. By adding a turn variable to a proposed mutual exclusion solution using a lock variable we can guarantee that the critical sections of several processes will be executed atomically. Which of the following is a problem with this type of solution?
   a. The solution will only work if a single processor is being used.
   b. Processes must enter their critical sections in strict alternation; one process will not be allowed to execute its critical section twice in a row, even if no other process is ready to execute.
   c. There are no problems with this solution.

72. Dekker presented the first recognized solution to the mutual exclusion problem. What hardware property is assumed by his solution?
   a. Write operations by multiple processors to the same memory location will be done atomically.
   b. Dekker’s solution requires that there be only a single processor.
   c. Multiple simultaneous read and write operations on the same memory location will be ordered so that all the writes occur first, then all the reads will take place.

73. Peterson’s solution to the mutual exclusion problem is much less cryptic than Dekker’s solution. It also uses a turn variable. What else distinguishes Peterson’s solution to the problem?
   a. His solution uses an array with one element per process to indicate which processes are interested in entering their critical sections.
   b. His solution works for all systems having at least one processor for each process attempting to enter their critical sections.
   c. None of the other choices characterize Peterson’s solution to the mutual exclusion problem.

74. Modern processors usually have special instructions that make implementation of solutions to the mutual exclusion problem much simpler. What is the name of an instruction found on IBM mainframe computer systems to support such operations?
   a. Probe and post
   b. Test and set
   c. Signal and set

75. The IBM mainframe instruction used to simplify the implementation of critical sections works by doing what?
   a. It eliminates the possibility of another process (or even another processor) examining the value of the control variable in memory between examining its value and then setting it to a known state.
   b. It shifts 1-bits out of a special shift register into a per-processor register, eliminating the possibility that two processors could perform the same operation simultaneously.
   c. None of the other choices correctly describe the operation of the IBM mainframe instruction.

76. What instruction on the Intel X86 family of processors could be used to achieve the same effect as that provided by the IBM mainframe instruction used to assist in the implementation of mutual exclusion?
   a. Add (ADD)
   b. Cache line flush (CLFUSH)
   c. Exchange (XCHG)

77. If processes have associated priorities, then a general rule is that
   a. the highest priority ready processes are those allowed to use the processors.
   b. the priority of a process is inversely related to the amount of memory it requires to execute.
   c. None of the other choices is a general rule about process priorities.

78. In some systems with priority-based processes, a problem can occur that results in the highest priority ready process being prevented from running by a lower priority process. This problem is called
   a. the inappropriate priority problem.
   b. the lowest priority problem.
   c. the priority inversion problem.
79. Suppose a low-priority process L holds a resource needed for the execution of a high-priority process H, but that L is prevented from running because a medium-priority process M is continually ready to run, and only a single processor is available. What is the name of an algorithm that could be used to deal with this problem?
   a. priority reversion
   b. priority adjustment
   c. aging

80. One technique that can be used to eliminating the busy waiting used with some solutions to the mutual exclusion problem is
   a. to add another processor.
   b. to limit solutions to a single processor system.
   c. to have a process relinquish the processor and block until another process, leaving its critical section, awakens it so it can repeat the test.

81. The processes involved in the producer-consumer problem are those that
   a. are compute-bound and I/O-bound.
   b. occasionally generate objects and those that then use those objects.
   c. are synchronous and asynchronous.

82. The objects that are being produced in the producer-consumer problem are generally characterized as
   a. data objects, like indications of keys pressed or released on a keyboard.
   b. static data items that cannot be altered by a process.
   c. blocks of memory, independent of their content.

83. The processes in the producer-consumer problem are assumed to run
   a. on the same single processor.
   b. at different and arbitrary rates, as is required by good solutions to race condition problems.
   c. at exactly the same rate.

84. A buffer, or storage area, is required in the producer-consumer problem. This buffer holds
   a. empty memory space that can be used by a consumer.
   b. objects that a consumer process considered, but then rejected as unsuitable.
   c. objects produced before a consumer process is ready to use them.

85. Access to the buffer in the producer-consumer problem
   a. is permitted only when the buffer is not completely full or completely empty.
   b. is shared by producers and consumers, and so must occur only inside a critical section.
   c. is controlled by the last object removed from the buffer.

86. When the buffer in the producer-consumer problem is completely full
   a. the size of the buffer is increased by fifty percent.
   b. a producer that has created an object must block until an empty space becomes available in the buffer.
   c. the number of producer processes is reduced by half.

87. When the buffer in the producer-consumer problem becomes empty
   a. consumer processes wishing to obtain an object must block until the producer places an object in the buffer.
   b. the speed with which producer processes are executed is increased.
   c. a dummy object is placed in the next buffer location that will be accessed by a consumer.
88. Assume a producer process is blocked while waiting on empty space in the buffer. In this case, a consumer process  
   a. change its priority to be at least as high as that of the blocked producer process.  
   b. will remove and discard sufficient data objects (from the buffer) to make space available for the blocked producer to place additional objects in the buffer.  
   c. will awaken a producer when it removes an object from the buffer, allowing space for the producer to place another object.

89. Consider that the statement `count = count + 1;` appears in multiple processes or threads in a system where the variable named `count` is accessible by multiple processes or threads. What must be done to ensure the value of `count` is correctly incremented when the statement is executed?  
   a. The variable `count` must never have a value other than 0 or 1.  
   b. The statement must be in a critical section in each of the processes or threads that contains it.  
   c. Each of the processes or threads must have the same priority.

90. Who proposed the semaphore data structure?  
   a. Donald Shell  
   b. Tony Hoare  
   c. Edgser Dijkstra

91. What is the significance of the letters `P` and `V` when describing the fundamental operations on a semaphore?  
   a. They correspond to the first letters of the Dutch words that describe the operations.  
   b. They correspond to the first letters of the words Permit and Validate, which describe the actions of the fundamental operations.  
   c. None of the other answer choices is correct.

92. The problem with the producer consumer problem solution using the sleep and wakeup functions is related to the fact that  
   a. a wakeup of a process (either a producer or a consumer) could be lost.  
   b. a wakeup may cause two processes to be awakened.  
   c. in some cases, sleep will cause two processes to block.

93. A semaphore has two data components. One of these is a set. What type of data objects can this set contain?  
   a. process identifications  
   b. processor affinities  
   c. None of the other answer choices is correct.

94. A semaphore has two data components. One of these is an integer. What does the value of this integer represent?  
   a. the number of waiting wakeups for processes that request use of a resource controlled by the semaphore  
   b. the number of processors utilizing resources controlled by the semaphore  
   c. None of the other answer choices is correct.

95. A `P` operation on a semaphore is also called a(n) _____ operation.  
   a. down  
   b. right  
   c. borrow

96. A `V` operation on a semaphore is also called a(n) _____ operation.  
   a. up  
   b. verify  
   c. down
97. When a P operation is executed on a semaphore with a count of zero,
   a. the process executing the P operation is blocked.
   b. the identity of one of the blocked processes in the set associated with the semaphore is removed from
      the set, and the corresponding process is moved to the ready state.
   c. the count is incremented and the process executing the P operation is blocked.

98. When a V operation is executed on a semaphore with a count of zero,
   a. if any processes are identified by the set associated with the semaphore, one of them is removed from
      the set and moved from the blocked to the ready state; otherwise the count is incremented by 1.
   b. the process executing the V operation is moved from the running to the blocked state, and its identity
      is added to the set associated with the semaphore.
   c. None of the other answer choices is correct.

99. Which of the semaphore operations, P or V, must be executed atomically?
   a. Neither P nor V must be executed atomically.
   b. Either all P or all V operations on a given semaphore must be executed atomically, but it makes no
      different which operation is selected for atomic operation.
   c. Both operations must be executed atomically.

100. In the semaphore-based solution to the producer/consumer problem, both the producer and the consumer
    processes use access to the shared buffer as a resource, and use it in a mutually-exclusive manner by
    performing down and up operations on a semaphore named \texttt{mutex}. What other resource do producer
    processes require if they are not going to be blocked?
    a. empty locations in the buffer into which a producer can place an item
    b. access to the variable containing a consumer’s process identification
    c. None of the other answer choices is correct.

101. Who proposed the synchronization structure called a \texttt{monitor}?
    a. Tony Hoare and Edgser Dijkstra
    b. Tony Hoare and Per Brinch Hansen
    c. Brian Kernighan and P. J. Plauger

102. A monitor is different from a semaphore or an event counter in several ways. A semaphore and an event
    counter are each data structures, but a monitor is
    a. a programming construct intended for inclusion in a programming language.
    b. an array of data structures.
    c. a set of threads, each of which can be used by a single process at a time.

103. It is the responsibility of the compiler that processes the programming language supporting monitors to
    guarantee that
    a. no more than one process or thread is executing inside multiple functions inside the monitor at one
        time.
    b. only one process (or thread) is allowed to execute any of the functions or the initialization code inside
        the monitor at one time.
    c. None of the other answer choices is correct.

104. A condition variable used with a monitor is similar to a semaphore in that it includes a set. It is different
    from a semaphore, however, in what way?
    a. There is no count associated with a condition variable.
    b. A condition variable includes two counts, one giving the number of resources in use, and one giving
       the number of processes waiting for a resource.
    c. The count associated with a condition variable may become negative.
105. A signal operation on a condition variable associated with a monitor, with no processes waiting on the condition variable,
   a. is ignored.
   b. causes a wakeup to be stored, so the next process that does a wait operation on the condition variable is not blocked.
   c. causes the process performing the signal operation to become blocked on the same condition variable that was signaled.

106. Brinch Hansen and Hoare each proposed a different way of dealing with a process P that signaled a condition variable and caused another process, say process Q, to be awakened. Why was it necessary to devise such strategies?
   a. The signaling process, P, is not allowed to continue execution inside the monitor while the awakened process, Q, is also executing inside the monitor.
   b. The two strategies differed depending on whether Q needed to execute immediately after being signaled, or if it could possibly wait until P finished executing.
   c. None of the other answer choices is correct.

107. Which of the following synchronization techniques can easily be used to support processes running on separate machines connected to a network?
   a. message passing
   b. semaphores
   c. Each of the above techniques can be used to synchronize processes on separate machines connected to a network.

108. When a process executes the appropriate code to receive a message, what will likely happen if a message is not immediately available for it to receive?
   a. The process will block until such time as a message is available.
   b. If a message intended for a different process is awaiting delivery, then it is given to the process for which no message was available; otherwise the process is blocked until such time as any message is available.
   c. None of the other answer choices is correct.

109. In the producer-consumer solution using message passing, there is some initialization required. What takes place during this one time only initialization?
   a. Each of the consumer processes is placed in the blocked state.
   b. An appropriate number of messages, each marked as being empty, are sent from the consumer to the producer.
   c. An appropriate number of messages, each marked as full but having arbitrary content, is sent by the producer to the consumer.

110. Which of the following statements about the equivalence of process synchronization techniques is true?
   a. Monitors and event counters can be shown to be equivalent.
   b. Each of the techniques - monitors, event counters, semaphores, and message passing - can be shown to be equivalent to each of the others.
   c. Message passing and semaphores can be shown to be equivalent.