Week 3 Practice Quiz

1. In the fragile queue library, can only the functions in that library change queue data?
   - No, the user can change not only the values in the queue but also in the queue structure
   - No, but the user can only change values in the queue structure; the library functions must be called to change values in the queue
   - No, but the user can only change values in the queue; the library functions must be called to change values in the queue structures
   - Yes, you have to call one of those functions to change the values in the queue or queue structure

Correct As the queue and queue structure are not declared private (or static), they are visible to the entire program.

2. As the queue structure in the robust library is a "dangerous implement", how does the caller refer to a queue?
   - The queue creation functions return a token encoding information about the queue, and that token is passed to library functions for queue operations.
   - The queue creation functions create a printable 4-character name for the queue, and that name is passed to library functions for queue operations.
   - The queue creation functions return the address of the queue, which is then passed to library functions for queue operations.
   - The queue creation functions return an index that refers to the queue, and that index is passed to library functions for queue operations.

Correct The queue creation functions provide information the caller can use to change the queue data or headers, doing what the "dangerous implements" designation is to prevent. The queue identifier is 32 bits in this example, but the characters are usually not printable.

3. In the robust library, what is the nonce used for?
   - To create a unique index for each queue
   - To enable a forged token to be detected
   - To distinguish a current queue from a deleted one with the same index
   - To count the number of queues created

Correct The nonce has nothing to do with the index or the number of queues created. Forged tokens are detected by the index and nonce not being decoded correctly. As the nonce is unique for each token created, and the token is stored in the queue head for validation, a deleted queue will have a different nonce number and cause an error.

4. In the robust library, the token generation function qtktref always returns a token with the high order bit (sign bit) 0. Why?

- A negative token means the function is called with a negative index
- A negative token will be ignored by the caller
- A negative token might be confused with an error indicator
- If the sign bit is 1, then the token cannot be 0

Correct Because of the structure of the calls to qtktref, that function would never be called with a negative index. A negative token does not mean the function is called with a negative index and a negative token will not be ignored by the caller, as all error codes are negative when a function returns a token - if it is negative it will be treated as a negative number.

5. In the robust library's take_off_queue function, after an if/else statement in which the else is unconditional (that is, not an else if), there is code to report an inconsistency that should never be reached. Which of the following is the best explanation for why is it there?
   - It is there to ensure that anyone who reads the code will know the author of the code programmed it robustly.
   - The compiler might compile the code incorrectly, and leave a way to reach that part of the code
   - The computer may execute something incorrectly and cause a branch to that part of the code
   - Someone might change the if/else in such a way that neither branch is taken, creating a way to reach that part of the code

Correct This describes a situation that may well occur, so it is clearly the best answer. If the computer is compiling or executing code incorrectly, then robust programming will not make the execution of the program robust.

6. Which of the following is *not* a problem with the put_on_queue() routine in the fragile library?
   - Its arguments are not checked for validity.
   - Queue underflow may occur.
   - The count field in the queue structure may not reflect the actual number of elements in the queue.
   - Queue overflow may occur.

Correct There is no check on the number of elements in the queue before the new one is added, but as elements are never removed from the queue, underflow cannot occur. That the count field in the queue structure may not reflect that actual number of elements in the queue discounts the ability of the count field to be changed externally to the library, which it can be.

7. Which of the following is not a problem with the qmanage() routine in the fragile library?
    - The value of size may be negative.
    - The size of the queue may vary among systems
    - The values of flag (the second parameter) and size (the third parameter) may be easily confused.
    - The address returned by the memory allocation routine is not checked, as virtual memory makes running out of memory very unlikely.

Correct While the size of the queue may vary among systems (the value can vary due to padding), it is completely irrelevant; that appropriate amount of memory is allocated for it. The other options all identify problems that need to be corrected, or require the routine or library be rewritten.