Process Synchronization

Task 1: Semaphore-based solution for producer-consumer problem

In the bounded-buffer problem, the producer produces information that is consumed by the consumer. Assume that the producer and the consumer are implemented as separated threads. The common buffer is an array and the producer and the consumer threads will move item to and from the buffer that is synchronized with a semaphore-based structure.

Internally, the buffer consists of a fixed-size array of type `buffer_item`. The array is manipulated as a circular queue with two functions, `insert_item()` and `remove_item()`, which are called by the producer and consumer threads, respectively. The definitions about the buffer and the two functions are declared in a header file (`buffer.h`).

The main function will initialize the buffer and create the separate producer and consumer threads. It will also sleep for a period of time after that and will terminate the application.

The main function will pass three parameters on the command line. They are the time of how long to sleep, the number of producer threads and the number of consumer threads. The skeleton for this task can be found in `skeleton_buffer.c`; where we have implemented the code for the `producer` and the `consumer` threads, and the function: `insert_item()` as well.

(a) Write the code for function `remove_item()` in the skeleton.
(b) Write the code for the main function in the skeleton and test it.

Exercise Questions:

Q1) Explain why spinlocks are not appropriate for single-processor systems yet are often used in multiprocessor systems.

Q2) Describe how the Swap() instruction can be used to provide mutual exclusion that satisfies the bounded-waiting requirement.