Project 0

Author: Xinpeng Wei

1. The main function contains calls to exit() (line 66) and pthread_exit() (line 80). How will the effect of these two calls differ when they are executed?
   Ans: `exit` will terminate the whole process including all the threads whereas `pthread_exit` will only terminate the current thread without affecting other threads and shared resources like mutexes.
2. The main function calls pthread_join() (line 77) with the parameter thread_return. Where does the value stored in thread_return come from when the consumer_thread is joined?
   Ans: the value comes from the return value of the consumer_thread, which is a pointer to the heap.
3. Where does the value stored in thread_return come from if the joined thread terminated by calling pthread_exit instead of finishing normally?
   Ans: then it will come from the parameter the joined thread passed in when calling `pthread_exit`.
4. On the same call to pthread_join() (line 77), what will it do if the thread being joined (consumer_thread, in this case) finishes before the main thread reaches that line of code (line 77)?
   Ans: then this call will return immediately.
5. In this program, the main thread calls pthread_join() on the threads it created. Could a different thread call pthread_join() on those threads instead? Could a thread call pthread_join() on the main thread (assuming it knew the main thread's thread ID - i.e. pthread_t)?
   Ans: Yes to both questions. According to the man page, all of the threads in a process are peers, which means that any thread can join with any other thread in the process.
6. The consumer_routine function calls sched_yield() (line 180) when there are no items in the queue. Why does it call sched_yield() instead of just continuing to check the queue for an item until one arrives?
   Ans: `sched_yield` allows the CPU to schedule other threads that currently can continue execution(e.g. producer pushing items into the queue) and not calling it will waste the CPU time but won't lead to any correctness issues.