Project 3
Alekhya Gade (ag13aj)

In main() function: The structure of the main function is implemented as given in the problem. Inputs are accepted using the command line arguments. Negative values are implicitly converted to positive values using absolute (abs). A buffer of BUFFER_SIZE is created. BUFFER_SIZE is defined in buffer.h. Based on the command line input required number of producers and consumers are created using pthread_create. Now as the producer and consumer threads are initialized, the next step is to set the main process to sleep.

Producer thread:

The producer thread creates a new item to be inserted into the buffer using the insert_item(...) method. Before the item is inserted into the buffer, the producer sleeps for a random amount of time. This is determined by rand % 10 generating sleep time ranging from 0 to 10 inclusively. An item based on rand() function is created and now the producer enters into critical section.

In the critical section, first we wait on empty semaphore previously declared and later a lock on mutex is applied. If thats returns successfully, the producer enters into critical section of writing to the actual buffer. Once the producer finishes its job, the lock on mutex will be released and a the full semaphore is signalled. The producer is looped over till the main function terminates and keeps inserting into the buffer.

Consumer thread:

Consumer is similar to the producer; except in this instead of inserting into the buffer, we remove an element(using remove_item(...)) from the buffer after performing necessary checks as we saw in producer's critical section.

Thread Safeness:

Thread safety is assured by the necessary check we apply before the producer or consumer enter into critical section(buffer or shared area) using a mutex. This producer never waits for consumer because as soon as the buffer is empty the producer threads insert the elements into the buffer and vice versa. Thus we avoid deadlocks between producer and consumer. At any point of the execution of the threads and the main process still not terminated, if the user wishes to terminate using the Ctrl+C, it generates a SIGINT trap which is caught by the main function and handled to sig_handler() function. The sig_handler() function now sets the kill_flag to 1 (initialized to 0 globally) and this means we are asking the producer and consumer threads to kill gracefully. This

is handled by producer and consumer loop check sequence. Every time the producer/consumer finishes, it checks for the kill_flag and if it is set, we break the loop and thus producer/consumer is terminated. By this time the, mutex lock released and the buffer is also in a safe state. The main process will terminate following this.

Sample output:

```
gcc -pthread -o semaphore semaphore.c
./semaphore 4 50 60

 ----------------Configuration ----------------------
 SLEEP TIME = 4
 NO. OF PRODUCERS = 50
 NO. OF CONSUMERS = 60
producer produced 1102520059
producer produced 1540383426
producer produced 278722862
consumer consumed 278722862
consumer consumed 1540383426
consumer consumed 1102520059
producer produced 1411549676
consumer consumed 1411549676
producer produced 1063958031
consumer consumed 1063958031
producer produced 2007905771
consumer consumed 2007905771
producer produced 1610120709
consumer consumed 1610120709
producer produced 498777856
consumer consumed 498777856
producer produced 327254586
producer produced 1572276965
consumer consumed 1572276965
^Cconsumer consumed 327254586
 Received Ctrl + C
 Asking producer and consumer to kill themselves
```