Consider the following producer-consumer problem. A producer process P forks to create a consumer process Q. The processes P and Q share a common buffer BUF[100] of 100 characters through which they exchange data. In each produce-consume iteration, the producer will write data into BUF and the consumer will read from it. We must make sure that (a) the consumer reads the data before the producer overwrites it with the data for the next iteration, and (b) the consumer does not read the same data twice. For this purpose, we need to synchronize the producer and the consumer.

After the initial steps, P and Q enter loops as shown below:

Production Loop for P
Begin Forever

Produce( );
Wake-up-consumer;
If (DONE) then break;
else Wait-for-consumer;

End Forever

Consumption Loop for Q
Begin Forever

Wait-for-producer;
Consume( );
If (DONE) then exit;
else Wake-up-producer;

End Forever

In each iteration, the producer reads a line from an input file `inp.txt` and writes it into the shared buffer BUF. It then wakes up the consumer and goes to sleep. The consumer (on being woken up) reads the line from BUF, prints it, wakes up the producer, and goes to sleep. When the producer reaches the end of `inp.txt`, it writes a null character in the first location of BUF, wakes up the consumer (for the last time) and exits. When the consumer finds that the first location of BUF contains a null character, it exits.

1. Write a main program which forks a child process and creates a shared buffer BUF[100] of 100 characters. Two semaphores X and Y are also created. Process P will wait on X when needed, and process Q will wait on Y when needed. The parent which acts as the producer P opens a file `inp.txt`.

2. Write the function Produce( ) which does the following. If it has reached the end of `inp.txt`, it will write a null character in BUF[0] and sets a flag DONE. Otherwise, it reads the next line from `inp.txt`, and writes it into BUF.

3. Write the function Consume( ) which does the following. If BUF[0] contains a null character, then it sets a flag DONE. Otherwise, it reads a line from BUF and prints it.

4. Write appropriate code using semaphores for the following statements:

Wake-up-consumer	:	Wakes up consumer
Wait-for-consumer	:	Puts producer to sleep on semaphore X
Wait-for-producer	:	Puts consumer to sleep on semaphore Y
Wake-up-producer	:	Wakes up producer

5. After the producer process P breaks out of its loop, it should wait for the consumer process Q (which was its child) to terminate. When this happens, the process P releases the shared memory and semaphores, and then terminates. Implement this within the `main()` function.

Submit a single C/C++ file solving all the above parts.
The file must contain your name and roll number.