```
/* Bounded-Buffer Producer/Consumer Problem using Semaphores */

#include <stdio.h>
#include <conio.h>

typedef struct Semaphore {
	int count;
} semaphore;

semaphore s, n, e;


int wait(semaphore *s) {

	(*s).count--;

	if((*s).count<0) {
		return 0;
	}

	return 1;
}

void signal(semaphore *s) {
	(*s).count++;
}

void produce() { printf("\tProducer produced an item.\n"); }
void append()  { printf("\tProducer appended an item.\n"); }
void take()    { printf("\tConsumer took an item.\n");         }
void consume() { printf("\tConsumer consumed an item.\n"); }

void producer() {
	printf("\n\nProducer:\n");
	produce();
	if(!wait(&e)) { printf("\tProducer cannot append because buffer is full.\n"); return; }
	if(!wait(&s)) { printf("\tProducer cannot append because buffer is locked.\n"); return; }
	append();
	signal(&s);
	signal(&n);
}


void consumer() {
	printf("\n\nConsumer:\n");
	if(!wait(&n)) { printf("\tConsumer cannot consume because buffer is empty.\n"); return; }
	if(!wait(&s)) { printf("\tConsumer cannot consume because buffer is locked.\n"); return;
}
	take();
	signal(&s);
	signal(&e);
	consume();
}


void main() {

	s.count = 1;
	n.count = 0;
	e.count = 3;

	clrscr();

	consumer(); getch();

	producer(); producer(); producer(); producer();

	consumer();

	getch();

}
```

OUTPUT

```
Consumer:
        Consumer cannot consume because buffer is empty.


Producer:
        Producer produced an item.
        Producer appended an item.


Producer:
        Producer produced an item.
        Producer appended an item.


Producer:
        Producer produced an item.
        Producer appended an item.


Producer:
        Producer produced an item.
        Producer cannot append because buffer is full.


Consumer:
        Consumer took an item.
        Consumer consumed an item.
```