```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <semaphore.h>

typedef struct{
    int size, insert_pos, extract_pos;
    sem_t empty, full;
    void ** buffer;
} Queue;

Queue * Queue_init(int N){
    Queue * q = malloc (sizeof(Queue) );
    q->size = N;
    q->buffer = malloc( N * sizeof(void*) );
    sem_init(&q->empty, 0, N);
    sem_init(&q->full, 0, 0);
    return q;
}

void Queue_destroy(Queue * q){
    free ( (*q).buffer);
    sem_destroy(&q->empty);
    sem_destroy(&q->full);
    free(q);
}

void Queue_insert(Queue * q, void * item){
    sem_wait(&(q->empty));
    q->buffer[q->insert_pos] = item;
    q->insert_pos =  (q->insert_pos + 1) % q->size;
    sem_post(&(q->full));
}

void * Queue_extract(Queue * q){
    void * item;
    sem_wait(&(q->full));
    item = q->buffer[q->extract_pos];
    q->extract_pos =  (q->extract_pos + 1) % q->size;
    sem_post(&(q->empty));
    return item;
}

Queue * q;

void * producer(void *arg){
    int i = 0;
    for (i = 0; i < 1000; i++){
        char * s = malloc(50*sizeof(char));
        sprintf(s, "Elem-%d", rand());
        Queue_insert(q, (void*) s );
    }
    printf("Inserted all. Waiting 1 second for consumer to extract.\n");
    sleep(1);
    exit(0);
}

void * consumer(void *arg){
    int i = 1;
    while (1){
        char * item = (char*) Queue_extract(q);
        printf("Extracted item %d: '%s'\n", i, item);
        free(item);
        i++;
    }
}

int main(int argc, char *argv[]){
    pthread_t t;
    q = Queue_init(10);
    pthread_create(&t, NULL, producer, NULL);
    consumer(NULL);
    Queue_destroy(q);
    return 0;
}
```