Process Execises

Exercise 1

Write a C implements the Producer Consumer architecture with limited buffer

Solution:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <semaphore.h>
#include <pthread.h>
#define N 10

int contatore = 0;
char * buffer [N];
pthread_mutex_t mutex;
sem_t empty, full;

void * producer(void *arg){
    int in = 0;
    while (1) {
        sem_wait(&empty); /* Attende che ci posto libero nel buffer */
        pthread_mutex_lock(&mutex); /* Acquisisce lock */


        buffer[in] = malloc(50*sizeof(char));
        sprintf(buffer[in], "Messaggio in posizione: %d", in);
        in = (in + 1) % N;
        contatore++;

        pthread_mutex_unlock(&mutex);
        sem_post(&full); /* Un dato un più nel buffer */

        sleep(1);
    }
}

void * consumer(void *arg){
    int out = 0;
    while (1) {
        sem_wait(&full); /* Attende che ci siano dati da consumare */
        pthread_mutex_lock(&mutex); /* Acquisisce lock */

        char * next_consumed = buffer[out];
        printf("Ricevuto: %s\n", next_consumed);
```

```
        free (next_consumed);
        out = (out + 1) % N;
        contatore--;

        pthread_mutex_unlock(&mutex);
        sem_post(&empty); /* Un posto libero in più nel buffer */
    }
}

int main(int argc, char *argv[]){

    pthread_t t;
    pthread_mutex_init(&mutex, NULL);
    sem_init(&empty, 0, N); /* Inizialmente N posti liberi */
    sem_init(&full, 0, 0);  /* e 0 occupati */
    pthread_create(&t, NULL, producer, NULL);
    consumer(NULL);
}
```