Operating Systems Lab
Lab-2

Date: 23-05-2021
Group 9
Abhishek Raj (180010002)
Harsh Raj (180010017)
Sri Priya (180010025)

Design Decisions For implementation:

zemaphore.h

It contains the declaration of struct zemaphore used for locking and unlocking.

```
 typedef struct zemaphore {
    int value;
    pthread_mutex_t mutex;
    pthread_cond_t cond;
} zem_t;
```

zemaphore.c

It contains implementation of various methods that operate of zemaphore. The two main implementations are Zem_down, and zem_up correspond to locking and unlocking of variables.

```
void zem_down(zem_t *s) {
    pthread_mutex_lock(&s->mutex);
    while(s->value <= 0)
        pthread_cond_wait(&s->cond, &s->mutex);
    s->value--;
    pthread_mutex_unlock(&s->mutex);
```

```
}
```

```
void zem_up(zem_t *s) {
    pthread_mutex_lock(&s->mutex);
    s->value++;
    pthread_cond_signal(&s->cond);
    pthread_mutex_unlock(&s->mutex);
}
```

test-toggle.c

This file prints the threads in sequential order. The sequential order is achieved by using the locking mechanism through struct zemaphore.

```
for(int i=0; i < NUM_ITER; i++) {
     zem_down(&zem[(NUM_THREADS+thread_id-1)%NUM_THREADS]);
     printf("This is thread %d\n", thread_id);
     zem_up(&zem[thread_id]);
}
```

Screenshots:

```
harshraj22 in zemaphore on main [$]
$ chmod +x test-zem.sh

harshraj22 in zemaphore on main [!$]
$ ./test-zem.sh
Running test-zem.c
This is main thread. This should print first
This is thread 1
One thread has printed
This is thread 2
Second thread has printed

Running test-toggle.c
This is thread 0
This is thread 1
This is thread 2
This is thread 0
This is thread 1
This is thread 2
This is thread 0
This is thread 1
This is thread 2
This is thread 0
This is thread 1
This is thread 2
This is thread 0
This is thread 1
This is thread 2
This is thread 0
This is thread 1
This is thread 2
This is thread 0
This is thread 1
This is thread 2
This is thread 0
This is thread 1
This is thread 2
This is thread 0
This is thread 1
This is thread 2
This is thread 0
This is thread 1
This is thread 2
```

The threads are printed in the sequential order, satisfying the expected order of execution.