Βιβλιοθήκη mysem.h

```
struct sem_t{
    int semid;
    int mtxid;
}

mysem_create{
    semid = semget();//create semaphore
    mtxid = semget();//create mutex for locking
}

mysem_destroy{
    semctl(semid, IPC_RMID);
    semctl(mtxid, IPC_RMID);
}
```

```
mysem_up{
    down(mtxid);
    //check with semctl(GETVAL) if semaphore
    //is already up, then:
        up(mtxid);
        return;
    //else:
        up(semid);
    up(mtxid);
}

mysem_down{
    down(semid);
}
```

Άσκηση 2(primes)

status	value
0	Not available
1	Available

sems	init
wait	0
available	0
work	1
mutex	1

```
struct thread_t{
        sems{
                wait, available, work, mutex}
        enum boolean term;
        int status;
        int *counter;
        int nthreads;
        pthread_t id;
        int number;
}
```

Worker_func

```
    while(1){
            down(wait);
            //check if term == true, then exit
            //primetest

            status = 1;
            down(mutex);
            counter++;
            //check if I am the very first available worker
            up(available); //wake up master thread if first available
            up(mutex);
            up(work);
    }
    up(work);
    Exit;
```

Master_thread

```
do{
        if(number <= 1) exit;
        down(mutex);
        //if no one available then down(available)
        up(mutex);
        down(available);
        //search for available workers(status = 1)
        //assign job → status = 0
        down(work);
        up(wait); //to unblock worker
        counter--;
        up(mutex);
}while(number > 1);

down(wait); //wait for workers to finish their jobs
term = true;
up(wait); //unblock them
######################################
down(work); //wait for workers to terminate
```

Άσκηση 3(bridge)

```
struct thread_t{
        sems{
                blueq, redq, mutex}
        enum colour_t colour;
        int max_cars;
        int curr_bridge;
        int red_waiting;
        int blue_waiting;
}
```

arrive_red

```
down(mutex);
//check if my colour == bridge colour{
        1)enough space on bridge
                curr_bridge++;
                up(redq); //if enough space
                up(mutex);
        2)max cars on bridge
                red_waiting++;
                up(mutex); down(redq);
}
//my colour != bridge colour{
        1)no one on bridge, blue waiting
                blue_waiting--; curr_bridge++;
                up(mutex); up(blueq);
        2)no one on bridge, not blue waiting
                colour = red; curr_bridge++;
                up(mutex);
        3)blue on bridge
                red_waiting++;
                up(mutex); down(redq);
                up(redq) //if there is enough space
}
```

sems	init
redq	0
blueq	0
mutex	1

red_func	blue_func
arrive_red	arrive_blue
//Crossing	//Crossing
leave_red	leave_bue

leave_red

```
down(mutex);
//full bridge{
        curr_bridge--;
        1)if red waiting
                red_waiting--; curr_bridge++;
                up(mutex); up(redq);
        2)up(mutex)
}
//I am the last one to exit{
        curr_bridge--;
        1)if opposite colour cars wait
                blue_waiting--; curr_bridge++;
                colour = blue;
                up(mutex); up(blueq);
        2)if my colour cars wait
                red_waiting--; curr_bridge++;
                up(mutex); up(redq);
        3)up(mutex)
}
//random leave{
        curr_bridge--;
        up(mutex);
}
```

Άσκηση 4(train)

```
struct thread_t{
        sems{
                ride, wait, out, in, finish, down,
                mutex}
        int max_pass;
        int curr_pass;
}
```

train

```
while(1){
        down(ride); //wait max_pass to enter
        for(max_pass times){
                up(wait); //pass enter train
                down(in); //wait each pass to enter
        }
//ride for 1 sec
        for(max_pass times){
                up(out); //wake each pass to exit the train
                down(down); //wait to excecute exit function
        }
        up(finish); //ride finished
}
```

sems	init
wait	0
ride	0
out	0
in	0
finish	1
down	0
mutex	1

passenger

```
down(mutex);
curr_pass++;
//check if max pass arrived{
        curr = 0;
        up(mutex);
        down(finish); //wait for prev ride to finish
        up(ride); //wake up train to start the ride
}
//else→ up(mutex);
down(wait); //wait to enter train
entering();
up(in); //in train
down(out); //wait for train to let me exit
Exiting();
up(down); //notify train that I have exited
```