1. Terminology (20p)

**race condition** occurs when two or more threads try to access shared data simultaneously, and the order in which they access the data can affect the result. This can be prevented by using semaphores to control access to the shared data.

**mutual exclusion** refers to the idea that only one thread can access a shared resource at a time. This can be implemented using semaphores, which allow only one thread to "acquire" the semaphore at a time, effectively excluding other threads from accessing the shared resource until the first thread "releases" the semaphore.

**starvation** refers to a situation where a thread is unable to acquire a resource that it needs because it is being constantly blocked by other threads. Semaphores can help prevent starvation by using a priority system, where higher priority threads are allowed to acquire the semaphore before lower priority threads.

**busy waiting** is a technique where a thread continuously checks if a resource is available, rather than blocking and waiting for the resource to become available. This can be inefficient and should be avoided, if possible, as it can waste system resources. Semaphores can be used to avoid busy waiting by allowing threads to block and wait for a resource to become available, rather than continuously checking if it is available.

# No-starve mutex (20p)

* 1. While thread 3 is waiting, thread 1 can keep signaling thread 2 and thread 2 can keep signaling thread 1. Thus, thread 3 is starving.

A thread that signals a semaphore while other threads are waiting, and then keeps running, waits on the same semaphore, and gets its own signal.

Imagine that Thread A gets the mutex and Thread B and C wait. When A leaves, B enters, but before B leaves, A loops around and joins C in the queue. When B leaves, there is no guarantee that C goes next. In fact, if A goes next, and B joins the queue, then we are back to the starting position, and we can repeat the cycle forever. C starves.

* 1. Before entering the critical section, a thread has to pass two turnstiles. These turnstiles divide the code into three “rooms”. The counters room1 and room2 keep track of the number of threads in each room. The counter room1 is protected by mutex in the usual way, but guard duty for room2 is split between t1 and t2. Similarly, responsibility for exclusive access to the critical section involves both t1 and t2. In order to enter the critical section, a thread has to hold one or the other, but not both. Then, before exiting, it gives up whichever one it has.

# Classroom (30p)

Constraints:

* The teacher unlocks the door of the class when there are 3 (or more) students present.
* student can leave during lecture
* teacher locks the door when no students are there.

Solution:

nr\_of\_students\_present = 0

classroom\_open = False

teacher = MySemaphore(0, "Teacher")

student = MySemaphore(0, "Student")

mutex = MySemaphore(1, "Mutex")

def teacherThread():

    while True:

        global nr\_of\_students\_present

        print("Teacher arrives")

        teacher.wait()

        mutex.wait()

        if nr\_of\_students\_present >= 3 and not classroom\_open:

            classroom\_open = True

            print("Open classroom")

            student.signal(nr\_of\_students\_present)

        mutex.signal()

        while nr\_of\_students\_present > 0:

            print("Gives lectures")

        classroom\_open = False

def studentThread():

    while True:

        global nr\_of\_students\_present

        mutex.wait()

        nr\_of\_students\_present += 1

        if nr\_of\_students\_present == 3 and not classroom\_open:

            teacher.signal()

        mutex.signal()

        if not classroom\_open:

            print("Waiting for classroom to be open")

            student.wait()

        print("Listen to lecture")

        mutex.wait()

        nr\_of\_students\_present -= 1

        mutex.signal()

# Synchronize (10p)

|  |  |  |  |
| --- | --- | --- | --- |
| **statement** | **turn** | **flag0** | **flag1** |
| C | 0 | False | False |
| D | “ | True | “ |
| T | “ | “ | “ |
| V | “ | “ | True |
| W | T1 stuck |  |  |
| E | T0 stuck |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |
| --- | --- |
| **statement** | **x** |
| J LOAD | 0 |
| N LOAD | 0 |
| J DEC | -1 |
| N DEC | -1 |
| J STORE | -1 |
| N STORE | -1 |
| K LOAD | -1 |
| K INC | 0 |
| K STORE | 0 |
| O LOAD | 0 |
| O INC | 1 |
| O STORE | 1 |
| --X8 | --X8 |
| P | 8 |
| Q | PRINT 8 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **statement** | **x** |
| J LOAD | 0 |
| N LOAD | 0 |
| J DEC | -1 |
| N DEC | -1 |
| J STORE | -1 |
| N STORE | -1 |
| K LOAD | -1 |
| K INC | 0 |
| K STORE | 0 |
| O LOAD | 0 |
| O INC | 1 |
| O STORE | 1 |
| --X9 | --X9 |
| J LOAD | 9 |
| N LOAD | 9 |
| J DEC | 8 |
| N DEC | 8 |
| J STORE | 8 |
| N STORE | 8 |
| K LOAD | 8 |
| K INC | 9 |
| K STORE | 9 |
| O LOAD | 9 |
| L | 9 |
| O INC | 10 |
| O STORE | 10 |
| M | PRINT 10 |
|  |  |

|  |  |
| --- | --- |
| **statement** | **x** |
| J LOAD | 0 |
| N LOAD | 0 |
| J DEC | -1 |
| N DEC | -1 |
| J STORE | -1 |
| N STORE | -1 |
| K LOAD | -1 |
| K INC | 0 |
| K STORE | 0 |
| O LOAD | 0 |
| O INC | 1 |
| O STORE | 1 |
| --X12 | --X12 |
| P | 12 |
| Q | PRINT 12 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **statement** | **x** |
| J | 9 |
| N | 8 |
| K | 9 |
| L | 9 |
| O | 10 |
| M | 10 |
|  |  |

|  |  |
| --- | --- |
| **statement** | **x** |
| J LOAD | 10 |
| N LOAD | 10 |
| J DEC | 9 |
| N DEC | 9 |
| J STORE | 9 |
| N STORE | 9 |
| K | 10 |
| O | 11 |
| L | 11 |
| M | 11 |
| P | 11 |
| Q | 11 |
| J LOAD | 11 |
| N LOAD | 11 |
| J DEC | 10 |
| N DEC | 10 |
| J STORE | 10 |
| N STORE | 10 |
| K | 11 |
| O | 12 |
| L | 12 |
| M | 12 |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **statement** | **x** |
| J | 9 |
| N | 8 |
| K LOAD | 8 |
| O LOAD | 8 |
| K INC | 9 |
| O INC | 9 |
| K STORE | 9 |
| O STORE | 9 |
| L | 9 |
| M | 9 |
| P | 9 |
| Q | 9 |
| J | 8 |
| N | 7 |
| K | 8 |
| L | 8 |
| M | 8 |
|  |  |