Lab 03 – Martino Mensio

Exercise 1

Buffer (queue)

The first functionalities to implement were the buffer management ones. In the program there are several queues: one shared normal queue, many urgent queues (one per office), one special queue, and many message queues (one for each student).

The student message queues are used to inform the students that an office has accepted them (beginning of service) and to inform that the service has ended (with relative information about what the student needs to do next).

The **send** and **receive** primitives of the Producers&Consumers are adapted for storing values of type **Info** instead of **int**.

When a thread needs to use the **send** or **receive** on the normal queue or on a urgent queue, it needs to acquire a mutex lock to update the value of the count of students in each of them. When the **send** is done on those queues, a broadcast message is sent on the condition variable, so that every office can check his condition to make his decision (take from normal or from urgent).

Main thread

The main thread, after checking and parsing the command line parameters, is responsible for initializing all the structures (semaphores, conditions, queues) and creating all of the threads.

In order to make the output more readable, the main thread prints out the current number of seconds elapsed at a constant rate (1 sec).

Student thread

Each student is a thread, that waits some seconds and enters the normal queue (**send** operation)**.** When an office decides to accept him, the student is notified via his message queue (doing a first **receive**). Then the student waits (doing a second **receive**) for a decision from the office about the need to go to the special office.

In this case, the student will do a **send** on the special queue and will wait again two messages on his queue (the first **receive** is an accepted message, the second one is the end of service from the special office).

After that, the student must go back to the normal office that previously accepted him (performing a **send** on the corresponding urgent queue) and wait again for two messages on his own queue.

Office thread

Each office is a thread, that has to look at two queues contemporarily: the normal queue (shared between all the offices) and his own urgent queue. Because the **receive** operation is blocking, and therefore it would be impossible to check both queues, the check is done on some counters (protected by a mutex) to see if some students are available in the queues. After checking the counters value, the office goes on a conditional wait (making the lock available for other offices), and is woken up by the broadcast performed when a student performs a **send** on one of the monitored queues. When waking up, the check on the counter is done again, and if the condition is satisfied the office can **receive** from one of the queues (his own urgent or the shared normal), with priority to the urgent one.

As soon as a student is accepted, the office sends him a message to make him know what's happening. After some time, the office decides (randomly with the specified probability) if the student can go home or if he needs to go to the special office (not possible if the student has already been there).

Special office thread

The special office work is simplified, because it performs **receive** only from his own queue, and does not need to use the condition variable mechanism.