Lab 3.1

Write a concurrent C program that simulates a university secretariat and a set of **`k`** students requiring service. The parameter **`k`** is given as an argument in the command line. Implement each student as a **thread.**

The secretariat has **`NUM_OFFICES`** "**normal**" offices to process the student files, and a "**special**" office in which a student goes if her/his file requires additional processing. Each office is a thread. All threads are created by the main thread, which then terminates.

A student enter the secretariat queue after sleeping a random time (between **`1`** and **`10`** seconds).
The student thread waits in a common queue, and is served by an available "normal" office.
The office threads take their next student from the shared "normal" queue.
The student thread prints its identifier (a number from **`0 to k-1`**) and the available office number (a number from **`0`** to **`NUM_OFFICES-1`**).
Ex.: **`student 0 received answer from office 1`**
The service lasts a random number of seconds (between **`3`** and **`6`**).
When served by the office, the student prints its identifier and the number of the serving office.
Ex.: **`student 0 terminated after service at office 1`**

The office clerk may decide (randomly, but **`40%`** of the times) that the student file needs additional information. It informs the student, that has to go to the "special" office, and to come back to the same office for further service. In this case, the student thread prints a string such as:
`"student 4 going from normal office to the special office\".`
When served by the special office the student prints a string such as:
`"student 4 served by the special office"`

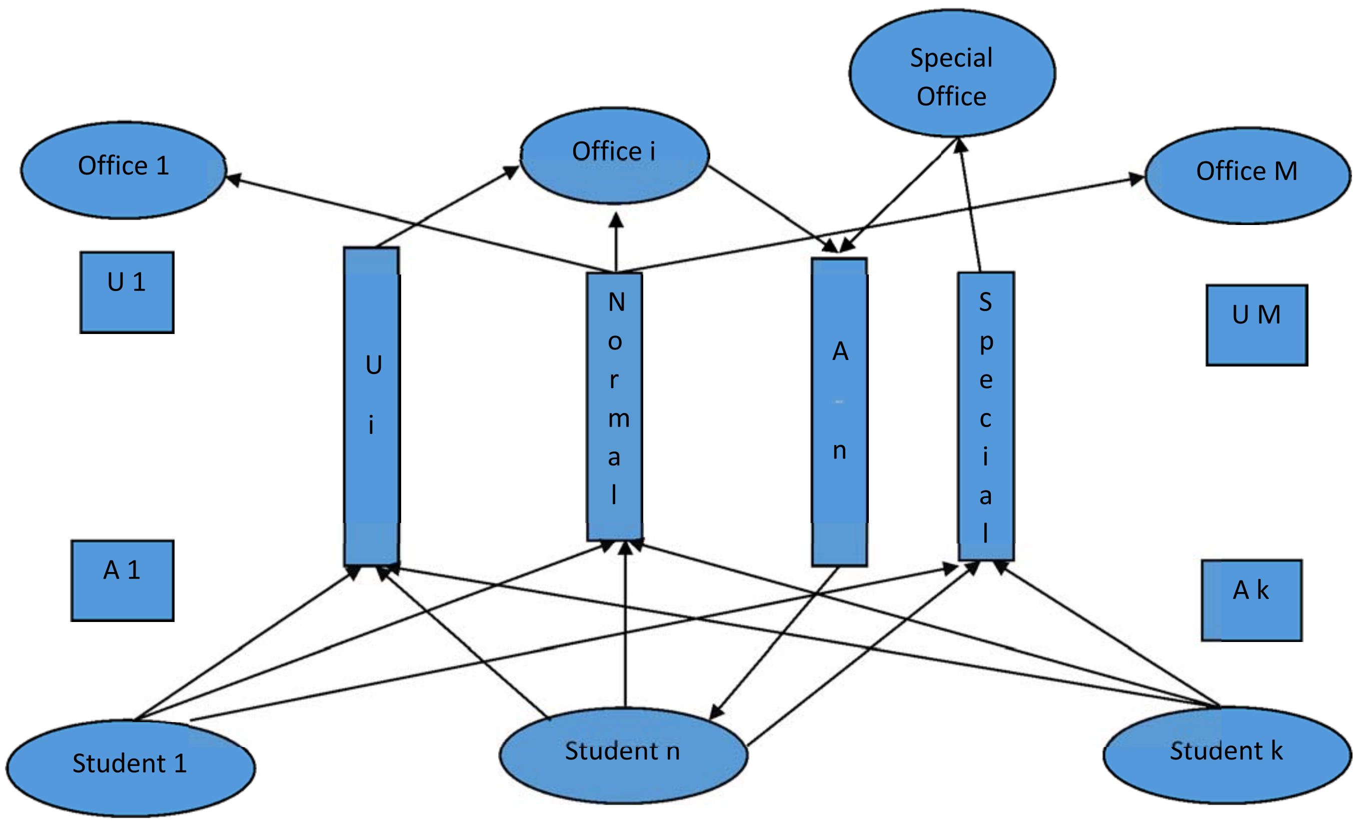
The office for special cases serves one student at a time in a random number of seconds (between **`3`** and **`6`**). After this service, the student comes back to the **previous** "normal" office, but **has precedence** with respect to the students waiting in the "normal" queue.
In this case, the thread prints a string such as: **`"student 2 going back to office 1".`**
The office service after special processing is faster: it lasts only **`1`** second.
When served again by the previous office, the student prints a string such as:
`"student 2 completed at office 1"`

In any case, a student thread finally prints a string such as:
`"student 3 terminated after service at office 0"`, then it exits.

When the last student has been served, all threads must terminate.

Hints:

- Use send and receive for inter-thread communication on shared buffers
- Use a condition to check if messages are present in the shared normal queue, or in the urgent queue of each office.
- The typedefs and global variables are given in **`office.h`**
- Use **`pthread_cond_broadcast`** rather than **`pthread_cond_signal`** to awake all offices waiting on the condition.



Create a compressed **`tgz`** archive file with this syntax:

Lab <LAB_number>_<Your_Id_number>_<YourLastname>. tgz

Example:

`Lab03_123456_Smith.tgz`

which includes all the source programs, and if necessary the input data, for each assignment.

For example: **`Lab3-123456_Smith.tgz`** must include **`Lab3.1.c`, and a description of your solution** and at least an **output log file**.

Drop your **`tgz file`** in the folder **Elaborati** of the SDP site of Portale della didattica.