You can find some clarifications and some warnings regarding the first homework below:

***How should we name our source file?***

Your main file (\*.cpp) **must have**the same name as your project folder => **XXXX-NameLastname where XXXX is your student number. Thus main file name must be XXXX-NameLastname.cpp. The other files, i.e., headers and other .cpp files, do not need to change.**

**What IDE do I use to write this and the following homework assignments?**

There is no restriction on how to develop the homework assignments. Use whatever text editor/compiler you like.  
However, the grader will use VS19 to grade your homework, so make sure your code compiles in VS19.

**Will the input files be placed in the root directory or will they be in a specific folder?**

In the homework, sample runs & test cases will be given as folders. These folders should be placed on the root of the project directory. You are required to prompt an input to take the folder name of the test case. Example folder structure as follows:

**“sample\_run\_1” folder in the root of your project directory:**

*< project\_folder >/sample\_run\_1*

**Configuration file in the “sample\_run\_1” folder:**

*< project\_folder >/sample\_run\_1/configuration.txt*

**Process #1 file in the “sample\_run\_1” folder:**

*< project\_folder >/sample\_run\_1/p1.txt*

**Process #2 file in the “sample\_run\_1” folder:**

*< project\_folder >/sample\_run\_1/p2.txt*

The number of process files is given in the configuration file, you should first read “configuration.txt”, then your program knows how many input file stream operations are required, which is the same as the number of processes.

After the execution of your program, you must create an output file in the “sample\_run\_1” folder:

*< project\_folder >/sample\_run\_1/output.txt*

\*Students can assume that every number in “configuration.txt” will be given as a positive integer, i.e., they all will be larger than zero.

***What is the order of adding elements to the top queue for Rule 5?***

The elements at the highest-ranked queue will not be moved. The other processes will be carried to the topmost queue starting from the next highest priority queue (if there are seven queues starting from the sixth) and ending with the least priority one (first queue). The queue with the highest priority is “Q< number\_of\_queues>”. The queue with the least priority is “Q1”. When moving elements from a queue to the highest priority one, first do a dequeue and do an enqueue to the highest priority queue.

**How are we going to initialize the MLFQ at the start of the program?**

In the initialization of the program, processes must be enqueued to the queue with the highest priority in increasing order, i.e., “P1” should be enqueued first, “P2” is the second (if exists), and so on.

**Can we use the standard queue class or do we need to implement our own queue class?**

Students are not necessarily required to implement a queue class, they can use the built-in queue class of C++.

**There is a contradiction between the output file example and sample run output file which one should we follow?**

There is a contradiction between the “output.txt” and in sample run in terms of representation of process, i.e., in the “output.txt”, it is shown as “P1”, however in the sample run, it is shown that as “PC1”. The format in the sample run is valid, thus students’ format should be the same as the format in the sample run.

#### In the sample output, why are processes being executed one less time than required? For example, process 4 should be executed 3 times, but it is executed two times then it is terminated. Is this a mistake?

**No, this is not a mistake.** This is because the last execution step of a process and its termination happen at the same time. In other words, in the final execution step of a process, the process is executed and then directly terminated and removed from the queues (i.e., moved to QX). For example, in the sample run given in the homework document, process 4 has execution steps 1, 0, and 1. Here are the lines related to it in the output file:

First, it is executed with a full slice (1):

1, PC4, Q3

The next time it is executed, it is executed with less than a time slice (0):

0, PC4, Q4

Finally, in its last execution, which is a full slice (1), we will execute it and directly terminate it:

E, PC4, QX

Note that the last step of a process will always terminate that process, regardless of whether the step requires a full time slice (1) or does not require a full time slice (0).