CSE 437 REAL TIME SYSTEM ARCHITECTURES

HOMEWORK 1 (Due 13th Nov. 2019)

Design a timer event generator in C++.

1. Design a thread-safe C++ class that implements the following interface. The class will have its own thread to provide the timing functionality. All the callbacks will be called from this single timer thread.

```
using CLOCK = std::chrono::high_resolution_clock;
using TTimerCallback = std::function<void()>;
using Millisecs = std::chrono::milliseconds;
using Timepoint = CLOCK::time_point;
using TPredicate = std::function<bool()>;

class ITimer {
public:
 // run the callback once at time point tp.
 virtual void registerTimer(const Timepoint &tp, const TTimerCallback &cb) = 0;
 // run the callback periodically forever. The first call will be executed as soon as this callback is registered.
 virtual void registerTimer(const Millisecs & period, const TTimerCallback &cb) = 0;
 // Run the callback periodically until time point tp. The first call will be executed as soon as this callback is
 //registered.
 virtual void registerTimer(const Timepoint &tp, const Millisecs & period, const TTimerCallback &cb) = 0;
 // Run the callback periodically. Before calling the callback every time, call the predicate first to check if the
 //termination criterion is satisfied. If the predicate returns false, stop calling the callback.
 virtual void registerTimer(const TPredicate &pred, const Millisecs & period, const TTimerCallback &cb) = 0;

};
```

2. Design a test application to test the class designed in (1)
3. Write a 2-page report containing descriptions of:
    a. The requirements of your timer, containing constraints and assumptions.
    b. The design of the timer
    c. How to build and test your project

Notes:

- The designed C++ project is going to be built both in Linux and Windows. To provide this feature, the homework should be built with CMake (Delivered homework should contain CMakeLists.txt).
- Required threading, timing and synchronization features should be implemented with native C++ (11, 14, …). No OS specific function calls are allowed (i.e. posix threads in linux, multimedia timer in Windows…)
- The resolution of the timer will be 5 milliseconds. If the callback is called outside 5 milliseconds' offset (earlier or late) from the registered time point, the timer will report "Deadline error", but the callback will still be called.
- A "good" solution will get the timer thread to block until either the timer queue changes, or a timeout occurs, rather than blocking with a constant period (Think what happens if registerTİmer is called while the thread is asleep).
- Your solution will be evaluated not only in terms of requirements, but also performance.
- The homework should be archived in a zip file (or a tar file) and uploaded to moodle. The archive should only contain the report, source files and CMakeLists.txt. It shouldn't contain compiled binaries.