ECE 3220 – Software Design in C & C++
Homework 3

In this homework you are expected to implement a multi-threading program in C++, which also handles starvation and race condition problems. This homework will cover the topics given below.

- Multi-threaded
- Handling race condition
- Handling starvation

You are expected to extend the the given reader-writer code, which is available on Canvas (readerwriter3.c).

- Your first task is to convert the C code into C++
- Current reader-writer code, handles the race condition problem, and also allows concurrency. However, when there are too many reader threads they dominate over the resource and no writer thread is available perform its task. The writer thread keeps on waiting for the lock to be released. This problem is called starvation. Your second task is to solve the starvation problem for both reader and writer threads.
- A couple of ideas to solve starvation:
  - Keep a counter for each thread, that counts the number of executions.
  - Specify an amount of reader threads that will enter the critical section. There will be no less or more threads that will enter the critical section. When that amount of threads has finished their job, you allow a writer in the critical section. However, this solution may not directly solve the starvation problem. On each turn you still might get the same writer running, or the same group of readers running.
  - Keep a prioritization number scale 1(low) – 5(high), and you assign the priorities to the threads. This will prevent a problem that you might encounter when you are a counter based approach (first solution). At some point your integer number might exceed and cause run-time problems. You can specify the prioritization algorithm as you like. For example, when a thread is executed you set its priority back to 1, but for each round the thread has waited you increase its priority score by +1. If a thread reaches already 5, you do not increment it anymore. You pick you threads that has the highest priority.
- While you try to solve the starvation problem do not turn your program into a sequentially running program, it still should be a concurrent program. Also, do not introduce any race condition problem.
- Write a single page report about how you tried to solve the starvation problem.
- You will have to use *pthreads* library, and to compile your code you need to use the *-pthread* option.

Submission rules:

1. Plagiarism is strictly prohibited. Any detected plagiarism will be graded as 0.
2. You Homework is due: **(04/22/2020)**
3. Late submission will not be allowed and graded as 0.
4. Include comments in your source code.
5. At total you must submit 3 files; a C++ file, starvation solution report, a Makefile.
6. Submit your files in the given format below;
   - C++ file: *hw3_studentID.c*. For example; *hw3_1234567.cpp*

- Makefile file: *Makefile*
- Report file: *hw3_report_1234567.doc/docx/odt.*

Grading criteria

1. (50 pts) Starvation solution
2. (10 pts) Proper usage of Comments
3. (10 pts) Handing race condition
4. (10 pts) C++ conversion
5. (20 pts) Report