INDEPENDENT STUDY PROPOSAL

CS 599: CONCURRENCY AND ALGORITHMS COURSE PROPOSAL

Student: Francis Joseph Serina Student Email: francis.serina@digipen.edu Class Level: MSCS

Student ID: 60001514 Term: Fall 2018 Credit Hours: 3

COURSE PURPOSE

To research and implement modern C++ features in Concurrency and apply them in Data Structures and Algorithms

GOALS AND OBJECTIVES

Through research and implementation, the student will gain practical knowledge about Multi-threading techniques used in the industry. Applying different techniques and scenarios will exercise the student's knowledge of parallel processes and thread management. Implementing parallel algorithms and data structures will test the capability of the student to determine which sections of the procedures could be parallelized.

MEANS OF COMMUNICATION

The student and adviser will communicate regularly via email and video calls weekly or bi-weekly to discuss progress, present learning notes, and share challenges encountered. The schedule for the meetings will be determined between the student and faculty member.

MEANS OF EVALUATION

The student may present bi-weekly presentations on Modern C++ Concurrency features. A presentation will focus on a feature with use-cases and sample executions. At least 4 presentations will be given. In addition, the student will implement at least 3 advanced algorithms using concurrency. See the Project Guidelines for details.

Grade	Explanation
A	Student displays strong command of the subject matter. Presentations and projects go above and beyond the expected requirements.
B	Student displays good understanding of the subject matter. Presentations are accurate and comprehensive. Projects faithfully implement the concurrency techniques and produce the expected results.
C	Student displays some understanding of the subject. Presentations and projects meet most but not all of the requirements.
D	Student displays little knowledge of the subject. Presentations and projects fail to meet most of the requirements.
F	Student displays no knowledge of the subject. Presentations and projects meet no requirements or are not even submitted.

GRADING POLICY

Projects:

Correctness	70%
Efficiency	20%
Readability	10%

Presentations:

Understanding of the Subject – Strengths, Limitations, Proper Usage	40%
Preparation – Slides, Demonstrations	30%
Presentation – Content Delivery, Time Management	30%

IMPLEMENTATION GUIDELINES

To gain practical knowledge of Concurrency Techniques, the student will implement at least 1 searching algorithm (Linear or Binary), 3 sorting algorithms (Quick Sort, Merge Sort, Heap Sort) along with the necessary data structures, at least 2 matrix operations (multiplication, inverse, linear solve), Histogram Equalization, and Fast Fourier Transform – all using modern concurrency techniques. The advisor may add additional requirements for the projects as they see fit.

COURSE PLANNING

The material for research will come from published materials from books or other sources. The following is a list of references that will be most valuable throughout the study:

- *Concurrency in Action*, Anthony Williams, 2012
- *Effective Modern C++,* Scott Meyers, 2014

More articles will be added to cover the main topics in Concurrency not covered by these. The advisor is free to recommend other materials or modify the current list as he or she sees fit.

SCHEDULE

The schedule is a weekly breakdown of tasks and topics. It is ultimately up to the discretion of the advisor if it should be modified. Also, the advisor may opt to have presentations organized per article rather than per topic for clarity.

Week	Tasks and Topics
1	Introduction to Concurrency in C++11

2	Demonstration: Parallel Multi-File Write
3	Presentation: Sharing data and Synchronizing Threads
4	Presentation: Memory Model and Atomics
5	Demonstration: Parallel Searching
6	Presentation: Thread-Safe Data Structures
7	Demonstration: Parallel Sorting
8	Presentation: Designing Concurrent Code, Thread Management
9	Demonstration: Matrix Operations
10	Research: C++17 Concurrency Features and Parallel Algorithms Library
11	Demonstration: Histogram Equalization
12	Demonstration: Fast Fourier Transform