Hanqi (Sheldon) Yang

(236) 965-4458 | hanqiyang1999@gmail.com | Victoria BC | Git: https://github.com/Sheldon-Y

TECH SKILLS

Programming: Python, C, C++, HTML, Java, AVR, SQL
Data Analysis: MATLAB, Excel
Web Development: HTML, CSS, JavaScript
Operating Systems: Linux, Windows, Mac OS
Productivity Tools: Office, Google Suite, Zoom, Teams
Software: Visual Studio Code, HBuilder, NetBeans, QT

EDUCATION

Bachelor of Science – Computer Science | University of Victoria **Apr 2022 – Sep 2025**

- **Relevant courses:**

Introduction to Computer Architecture	Algorithms and Data Structures	Operating Systems
Requirements Engineering	Database Systems	Quantum Algorithms

PROJECTS

Drawing Board | Personal Project **Feb 2024 -Present**

- **Development Tools**: Designed and implemented a drawing board application using the QT framework in C++, enabling freehand drawing and the creation of various geometric shapes.
- **Drawing function implementation**: Features Bresenham and Bezier algorithms for precise line and curve drawing, matrix operations for shape transformations, Cohen-Sutherland algorithm for clipping, and file operations for BMP, JPG, and PNG formats.

Real-Time Ship Health & Maintenance System Requirements Document | Course Project **Sep 2023 – Dec 2023**

- **Project Management**: Created and updated as part of a team, project management document including scope, revisions, glossary of terms, and diagrams for ensuring reliability, safety, and optimal performance.

Simple Shell Interpreter | Course Project **Sep 2023 – Dec 2023**

- **Command Parsing and Execution**: Applied C language in the Linux system to implement a shell through fork() and exec() functions, which can enter or exit the folder or change directories through Linux command and print out all the file names of the current folder.
- **Background Execution**: Capable to execute multiple processes in a folder and display the process status.

Simple File System | Course Project **Sep 2023 – Dec 2023**

- **System Design and Operations**: Constructed the file system structure featuring superblocks for global data, anodes for file metadata, and data blocks for storage. Implemented bitmap indexing for efficient free space management, using algorithms to allocate and deallocate space dynamically.
- **Directory and System Management**: Facilitated file and directory operations through specialized functions like create_inode(), write_file(), and read_file(), ensuring robust handling. Utilized system calls like mount() and unmount() for integrating the file system with the host operating system, incorporating real-time updates to superblocks upon session completion.