Concentrating on Computer Science at the University of Waterloo over the past three years, I discovered a keen interest in System areas, including Operating Systems, Concurrency, Compilers, etc. This was partly due to a course: Foundations of Sequential Programming (Enriched). Professor Ondřej Lhoták's announcement that we would build a feature-rich functional programming language compiler simultaneously excited and surprised me - it would be the first programming language I developed myself. We first developed an assembler to translate assembly codes to binary on a MIPS emulator. Then, we built highly abstract intermediate representations, including if statements, procedure calls, and closures. After that, we completed the front end of our compiler: converting the source code to an abstract syntax tree by CYK parsing and type checking. Finally, we combined the front and back ends of our compiler with the garbage collector by Cheney's Copying Algorithm as my first integrated compiler.

After tasting the accomplishment of building a complete compiler, I felt exceedingly confident about my capability in Computer Science, even a bit complacent. However, when I began my first Co-op term as a Research&Development Intern at Momenta, I was frustrated by the gap between the workplace and the campus. There were numerous tools and programming languages that I needed to be familiar with, such as Git, Docker, Python, Bash, Ros, Pytorch, etc. Fortunately, my supervisor toured me through all the prerequisite knowledge step-by-step. With his help, I developed a pipeline to extract training data from Rosbags, train deep learning models for autonomous driving cars, and validate and improve our models, and I also drafted the patent of the fusion algorithm our team worked for - The fusion method and device for multi-sensor data. Leaving Momenta, I joined Baidu's Recommender System R&D team for my next work term, mainly responsible for database management. These two Co-op experiences exposed to me two brand-new areas: Deep Learning and Database. More importantly, I also utilized novel systems that significantly enhance software's extensibility and maintainability for tech companies, effectively boosting software development efficiency. For instance, Baidu developed its own compiler on a distributed system with powerful computational capability dedicated to compiling, which merely takes several minutes to compile a colossal module. Therefore, compared to high-level areas, the underlying systems like Operating Systems and Compilers attracted me more deeply, which solidified my focus on the System areas of Computer Science during my undergraduate study.

In the Spring of 2022, I experienced system-level programming with hardware for the first time in the infamous "Trains" Real-time Programming course at the University of Waterloo. My teammate and I implemented a Real-Time MicroKernel from scratch on an ARM processor in C and ARM assembly: we wrote the context switch for software and hardware interrupts, designed our task structure, developed multiple syscalls, and built the communication primitives between tasks via the Send-Receive-Reply mechanism. Then, we utilized the Server-Worker pattern to develop various kernel drivers, such as timer driver and serial port driver, to offer users I/O and timing functionalities by hardware interrupts. After that, we built a train control system on the kernel to manipulate multiple electrical trains on the track, intending to ensure each train travels to specific spots precisely. To avoid collisions between trains, we established a real-time position server to query each train's position and designed a system to detect possible collisions to re-route

each train. Completing such a colossal project was challenging because every single component of this project, from the kernel to drivers to userspace programs, was designed and implemented by ourselves. One tiny design mistake at the beginning may cause hundreds of lines of code to be modified and a complete redesign. Eventually, when these trains traveled gracefully on the track, I felt extremely fulfilled. This experience confirmed my desire to pursue future research in System areas.

During my last two terms at the University of Waterloo, I also participated in two research projects as an Undergraduate Research Assistant to pursue in-depth study in Systems. I first worked with Prof. Martin Karsten to find a solution to embed his user-level thread library, Libfibre, into C++ such that std::thread will invoke high-performance user-level threads provided by Libfibre instead of heavy Pthread. After thoroughly investigating the GCC source code, I implemented a glue layer between std::thread and Libfibre to achieve this. Following several failures and debugging, I succeeded, and my name was put on the contribution list of Libfibre. Then, I worked as a research assistant to Prof. Peter Buhr. Similarly, this time I did a Pthread emulation research project to emulate Pthread routines by user-level threads in Cforall, an extension of C. I utilized interposition to preserve the functionality of the original Pthread to construct the runtime system and provide users with light Pthread routines empowered with user-level threading. These experiences allowed me to gain insight into cutting-edge academic research in Systems.

After carefully considering my previous academic and professional experience, I decided to pursue a Master's Degree in Computer Science and focus on Operating Systems and Compilers. I expect to receive more in-depth training by taking advanced courses and doing an internship to apply my theoretical knowledge and skills to solve real-world problems. The Master of Science in Computer Science programs in top U.S. universities specializing in Computer Systems match my goals exactly. I can gain a competitive edge in Systems and obtain highly sought-after skills for employers through advanced courses, including advanced Operating Systems, advanced Compilers and etc. More importantly, I will make good use of the university's well-established reputation across industries and close ties with renowned enterprises to gain practical experience through internships before graduation. A candidate with both theoretical competence and hands-on skills will have more opportunities in the job market.

In the short run, upon completing my Master's Degree, I plan to enter companies excelling at Compilers and Operating systems, such as Intel and Apple, and work as an R&D engineer dedicated to improving compiler optimization and solving OS-related issues like deadlocks. In the long run, I hope to organize a research team dedicated to developing a new programming language that can solve problems bewildering modern programming languages. In this way, I can convert what I have learned in Computer Systems over my undergraduate and graduate study into the impetus of modern technology.