NSERC Undergraduate Student Research Award
2018

For further information:

Scholarships are valued at $7840 (includes $4500 from NSERC and minimum $3340 from the faculty supervisor)

The Department of Computing and Software is accepting applications for the following projects. Applications must be received in the CAS Departmental Office (ITB/202) no later than Friday, February 9, 2018. Applications must include Part 1 of Form 202 (Application for an Undergraduate Student Research Award), available on the NSERC website, and official transcripts. The form must be completed electronically by logging into the NSERC website, then printed and signed prior to submission to the departmental office. The departmental submission should include a separate note indicating which project(s) you are interested in.

**Project # 1 (S. Smith and J. Carette): A New Approach to Software Development**

Interested in novel approaches to software design? If so, then you’re the person we’re looking for! Our team is testing a new approach to software development centered on using ‘recipes’ to assemble knowledge from a central repository to form all required software artifacts, from requirements documents to tests and, of course, code.

We have several examples, but they need to be rebuilt using our approach. And we need help to do this.

Main requirement: a willingness to learn new ideas and new programming languages. Our main infrastructure is in Haskell, we generate code in Python, C++, Java, Lua (and more, and trying to add more), as well as documentation in LaTeX and HTML. We also have several embedded DSLs.

All our work is publicly available: https://github.com/JacquesCarette/Drasil. Your contributions would then become part of your portfolio for future employers or graduate supervisors.

**Project # 2 (E. Sekerinski): Software Development for Water Quality Sensor Networks**

Two summer positions are available for a collaborative project that involves a local industrial partner as well as several communities in Canada that are affected by poor water quality. This project uses small low-power wireless sensor nodes (Arduino, Raspberry Pi) to which various water quality sensors (e.g. conductivity, turbidity, dissolved oxygen) are attached (some sensors are commercial and some are developed by other research groups at McMaster). The research aspects of the project are code generation from models (as well as correctness and quantitative analysis from the models) and using machine learning for analyzing the data (e.g. detecting deterioration of the sensors, abnormal events). The tasks involve developing software for visualizing and transmitting the data on nodes, interfacing of sensors, interfacing to a database and web server. While the task will be focused, this summer job allows to get rather broad experience and interact with other summer and graduate students and well as other researchers and users.

Required qualifications include a strong programming background and the willingness to catch up with
various technologies used. Experience with hardware interfacing, Arduino, .NET, databases, web sites is welcome.

**Project # 3 (D. Down): Quantifying the effects of cooling decisions on temperature distributions in a data centre**

Data centres have multiple cooling units that can be responsible for as much as 50% of total power consumption. This stems from the lack of a central controller that controls the multiple units. In this project we want to design a control algorithm that will coordinate the operation of multiple cooling units to minimize a data centre’s total power consumption. In order to be able to do that we need to understand the effect of cooling decisions on temperature distribution inside a data centre. The applicant will be responsible for running multiple experiments on our experimental data centre, the results of which will be used to develop a predictive model using machine learning techniques. The model would determine, given the current temperature distribution and a cooling decision (increase or decrease in set point or fan speed), the temperature distribution at a given time horizon. Ideal applicants will be familiar with machine learning concepts such as classification, regression, principle component analysis (PCA) and feature extraction.

**Project # 4 (W. Farmer): HOL Light QE**

Philosophers, logicians, and computer scientists have long been interested in metareasoning, that is, reasoning about reasoning itself. Reasoning is performed in a formal logic by manipulating symbols. Hence metareasoning in a formal logic involves reasoning about the manipulation of symbols. Traditional logics like first-order logic and simple type theory (a classical form of higher-order logic) do not provide much support for reasoning about symbol manipulation. To overcome this deficiency, we have developed a version of simple type theory called STTqe that includes quotation and evaluation operators like the quote and eval operators in the Lisp programming language. With these operators it is possible to reason about the interplay of the manipulation of symbols and what the manipulations mean mathematically.

HOL Light is a software system developed by John Harrison at the Intel Corporation that assists users in proving mathematical conjectures in a version of simple type theory. HOL Light is open source software written in OCaml. Although it is a relatively small system, it has been used to formalize many kinds of mathematics and to proof-check many proofs including the lion’s share of Tom Hale's proof of the Kepler conjecture.

This project is a continuation of a project started last year to test out the ideas embodied in STTqe by producing a modified version of HOL Light called HOL Light QE that includes quotation and evaluation operators. Last year's project was very successful: it produced a fully working HOL Light QE system. This year's project is to continue the development of HOL Light QE and to demonstrate its expressive power by proving the correctness of syntax-based mathematical algorithms like symbolic differentiation.

**Qualifications:**

A solid background in computer science or software engineering is required for this project as well as a strong interest in mathematics, logic, and formal proofs. Knowledge of functional programming and higher-order logic would be considered a significant plus.
**Project #5 (W. Kahl): Extending the Teaching Tool CalcCheck**

CalcCheck ([http://CalcCheck.McMaster.ca/](http://CalcCheck.McMaster.ca/)) is a proof checker for calculational proofs in the logics of the popular textbook "A Logical Approach to Discrete Math" (LADM) by David Gries and Fred Schneider currently used in COMPSCI&SFWRENG 2DM3.

CalcCheck is implemented in the functional programming language Haskell, with the web application aspects implemented in Haste. Although the system is already useful in its current state, there is ample room for improving the capabilities and the user experience, which is what this project plans to address.

This project requires strength and interest in logics and discrete mathematics. Some previous knowledge of Haskell is necessary.

The student's role will be to add functionality to CalcCheck, initially including at least some of server logging and access control, web interface improvements, automated exercise and test problem generation. In each case, this involves all of design, implementation, and documentation. After familiarisation with the system via these smaller projects, larger projects currently envisaged include support for checking program correctness proofs, and improved integration with Avenue conforming to the LTI standards.

CalcCheck: [http://CalcCheck.McMaster.ca/](http://CalcCheck.McMaster.ca/)

Haskell: [http://haskell.org/](http://haskell.org/)
Haste: [https://haste-lang.org/](https://haste-lang.org/)

**Project # 6 (W. Kahl): Utility Libraries for Certified Programming in Agda**

The dependently-typed programming language and proof checker Agda2 allows us to write functional programs that include their correctness proofs as part of the source code.

As part of an ongoing effort aiming to produce verified Agda implementations of general graph transformation mechanisms, this project will produce self-contained auxiliary libraries, including input and output facilities, file handling and data interchange, graph layout and display.

This project requires strength and interest in logics and discrete mathematics. Some previous knowledge of Agda or Haskell is necessary.

The student's role will be to produce libraries satisfying specific project needs. In each case, this involves all of design, implementation, and documentation.

Project home: [http://relmics.mcmaster.ca/RATH-Agda/](http://relmics.mcmaster.ca/RATH-Agda/)


**Project # 7 (D. Stebila): Post-Quantum Cryptography**

My research is in cryptography, especially the areas of public key cryptography and cryptographic protocols, which aim to build and analyze useful encryption algorithms from mathematical problems in number theory and algebra. For example, a key component of Internet security today is based on the RSA
algorithm, the security of which is based on the difficulty of factoring large numbers. An active area of research today is post-quantum cryptography, which aims to build cryptography on alternative mathematical problems, such as those involving lattices or error-correcting codes.

Projects can take a theoretical or applied angle (anywhere from proving theorems to writing software). The exact topic will be decided once we meet and discuss your interests. Students working with me on a USRA over the 16-week summer term would meet with me on a regular basis, and participate in my research group’s activities, such as a bi-weekly reading group with my graduate students and post-docs.

Minimum course requirements: basic discrete mathematics (CS 2DM3, CS 2FA3)

Preferable knowledge: some cryptography or information security (MATH 3CY3, CS 3IS3, CS 4C03), more number theory, or higher algebra (MATH 3H03, MATH 3E03). Knowledge of C and Unix is preferable for programming projects, but other languages might also be possible for software projects.