

**NSERC
2012
Undergraduate Student
Research Award**

For further information:

http://www.nserc-crsng.gc.ca/Students-Etudiants/UG-PC/USRA-BRPC_eng.asp

Scholarships are valued as follows (amount shown includes \$4500 from NSERC):

Students who have completed Level 1: \$5625
Students who have completed Level 2: \$7300
Students who have completed Level 3: \$8100
Students who have completed Level 4 or 5: \$8500

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 Tuesday, February 7, 2012. 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. Please also include a brief separate statement indicating which project(s) you are applying for. We do not guarantee that a candidate will be selected for every project.

- Project #1: Analysis of Student Grades and Other Metrics (S. Smith)**
- Project # 2: Implementation of a Family of Material Models Using Automatic Differentiation (S. Smith)**
- Project # 3: iOS Developer (C. Anand)**
- Project # 4: Coconut Developer (C. Anand)**
- Project #5: Games and Dialogue (J. Carette and W. Farmer)**
- Project # 6: Scale and Games (J. Carette and W. Farmer)**
- Project # 7: Extending the Teaching Tool CalcCheck (W. Kahl)**
- Project # 8: Verified Container Libraries in Agda (W. Kahl)**
- Project # 9: Verified Graph Transformation in Agda (W. Kahl)**
- Project # 10: FPGA Based Implementation of Verifiably Correct Safety Critical Systems (M. Lawford)**

Project #1: Analysis of Student Grades and Other Metrics (S. Smith)

Project Description:

Analysis of student data, including grades, Avenue activity, survey responses, and attendance, to determine meaningful predictors of students at risk of failure. This analysis will assist with understanding and hopefully improving student retention in Engineering 1 and in subsequent years. The project will consist of statistical analysis, including developing a Support Vector Machine (SVM). The SVM will analyse the data with the goal of early recognition of the patterns exhibited by at risk students. The analysis will be facilitated by previously developed database software called the Student Grades and other Metrics Database (SGMD). SGMD is a tool used for collecting student data, anonymizing the data, performing queries and creating reports.

Qualifications:

Software development and programming skills. Strong in mathematics.

Project # 2: Implementation of a Family of Material Models Using Automatic Differentiation (S. Smith)

Project Description:

The purpose of this project is to compare automatic differentiation versus symbolic processing and code generation for implementing a virtual material testing laboratory. Previously a 3D virtual material testing laboratory was developed with the material models generated using Maple's symbol manipulation and code generation features. This project will look at the advantages and disadvantages of using automatic differentiation for the same purpose. As for the previous virtual lab, the new lab should accommodate many different material tests including uniaxial, biaxial, multiaxial extension and compression, and shear tests in any direction. Material types that should be supported include elastic, viscous, shear-thinning, shear-thickening, strain hardening, viscoelastic, viscoplastic and plastic.

Qualifications:

Software development and programming skills. Strong in mathematics and physics.

Project # 3: iOS Developer (C. Anand)

Developer to support two different research projects:

- 1. development of electronic document workflow support on the iPad (joint project with local company who is always hiring)

2. development of interactive educational applications to support teaching either basic CS (ie 1MA3), optimization (ie 4TE3), or MRI (ie CAS750)

You should have an interesting iOS project, or strong skills in other software development, and creative ideas in one of the application areas. Some knowledge of Haskell would also be helpful.

Project #4: Coconut Developer (C. Anand)

Developer of compiler optimization technology to extend the functionality of Coconut to better exploit multi-core parallelism, based on ExSSP (see MSc thesis of Wolfgang Thaller) and pattern matching in directed acyclic graphs. You must know how to program in Haskell.

Project #5: Games and Dialogue (J. Carette and W. Farmer)

Abstract:

Dialogue in games is currently 'scripted' in a way which is quite similar to a movie, often using tools such as Word. This makes the asset thus developed very hard to re-use in the game itself, and these must be 'transcribed' into other formats. Not only is this very inefficient, it is also error prone, as well as making changes very difficult. Furthermore, such scripts contain all sorts of other information which needs to manually be checked -- like coverage, asset assembly, consistency, etc. This information could also be reused, if it were available in a more semantically meaningful way.

In conjunction with a local gaming company, we have been investigating a Domain Specific Language of game dialogues that would bridge this gap between script-writing by staff writers and the requirements of the technical team for semantically meaningful, digital assets they can re-use -- in such a way that changes by the technical writes could be smoothly and with minimum effort be incorporated into a game without requiring hands-on manipulation by programmers. There is a pre-existing prototype for a fragment of a script language, which needs to be completed, with respect to the much more complete requirements which we now have.

Solid programming skills and the willingness to learn new programming languages are required.

Project # 6: Scale and Games (J. Carette and W. Farmer)

Description:

Games are now played on devices of wildly varying size: from the wall-of-screens seen in ITB 225 to iPods and smartphones. Naturally, one cannot use the exact same game at all scales, but

one can nevertheless be remarkably uniform - as witnessed say by Plants vs Zombies. Nevertheless, some things must change.

This project consists of two parts: 1. continuing the development of a "language of scale", which describes constraints associate to screen size, and 2. applying this language to component specialization (like status display). As screen size varies, different real estate can be used for 'status display'; as screen real estate gets cramped, decisions must be made to restrict (and potentially eliminate) all permanent display of current status. The crux of the project is to understand what are the main decision factors involved, and to continue the development of an existing prototype, by adding more components which can be specialized.

Solid programming skills and the willingness to learn new programming languages are required.

Project # 7: Extending the Teaching Tool CalcCheck (W. Kahl)

CalcCheck 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 COMP SCI 1FC3.

CalcCheck is implemented in the functional programming language Haskell.

This project will add functionality to CalcCheck, possibly including:

- * Automated problem generation
- * Matching problems and solution attempts
- * Theory management
- * Support for checking program correctness proofs
- * WWW interface

Some previous knowledge of Haskell and LaTeX is required.

CalcCheck: <http://CalcCheck.McMaster.ca/>

Haskell: <http://haskell.org/>

Project # 8: Verified Container Libraries in Agda (W. Kahl)

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, in particular verified implementations of container data structures.

This project requires strength and interest in logics and discrete mathematics. Previous knowledge of Agda or Haskell would be an asset.

Agda: <http://wiki.portal.chalmers.se/agda/>

Project # 9: Verified Graph Transformation in Agda (W. Kahl)

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.

This project will be part of an ongoing effort aiming to produce verified Agda implementations of general graph transformation and also of more specific term graph transformation mechanisms that will be used in optimised machine code generation.

This project will produce verified implementations of selected transformation mechanisms, and expand the underlying theories as necessary.

This project requires strength and interest in logics and discrete mathematics. Previous knowledge of Agda or Haskell would be an asset.

Agda: <http://wiki.portal.chalmers.se/agda/>

Base theories: <http://relmics.mcmaster.ca/~kahl/RATH/Agda/>

Project # 10: FPGA Based Implementation of Verifiably Correct Safety Critical Systems (M. Lawford)

This project involves investigation of the design and implementation of safety critical systems for the nuclear industry using FPGA based platforms. The focus will be upon construction of pre-verified functional blocks that can be used to implement verifiably correct systems.

The candidate should have experience with FPGAs, embedded systems design and knowledge of Verilog and/or VHDL. Experience with related CAD tools is considered an asset.