McMaster University	**COMP SCI 3EA3**
Department of Computing and Software	**Outline**
M. Al-hassy	**2017-01-01**

Software Specification and Correctness

COMP SCI 3EA3

Instructor: Musa **Al-hassy**, Department of Computing and Software, ITB-229
E-Mail: alhassm@mcmaster.ca

Calendar Description:

Formal specifications in software development; logical formalisms; functional and relational specifications; completeness and consistency of specifications; verification; validation; presentation of information; tool supported verification.

Goals:

- Understanding of the motivation of mathematical approaches to software specification
- Ability to produce and evaluate formal software specifications
- Knowledge of one typical approach to formal software specification and verification
- Appropriate uses of assertions and an understanding of their limitations.
- *Derivation of correct programs after formalising their English prose specifications.*
- Familiarity with certified and verified programming.

Course Page: `http://www.cas.mcmaster.ca/~alhassm`

The course pages will contain useful links to external material, and will also serve as central location for making information and material available including an up to date schedule of what is being covered, including slides. It is the student's responsibility to be aware of the information on the course web page, and to check regularly for announcements (or RSS subscribe).

Schedule:

Lectures:	BSB-105; 8:30-9:30 Mondays & Wednesdays; 10:30-11:30 Fridays.
Tutorials:	BSB-B154; 14:30-15:30 Tuesdays

Classes and tutorials start on the 4th and 9th of January, respectively.

Occasionally, special sessions in computer-equipped labs may be announced.

Students are expected to attend all lectures and tutorials.

Textbook

"Backhouse": Roland Backhouse: **Program Construction Calculating Implementations from Specifications**, Wiley 2003, ISBN 0-470-84882-0

This book may be construed as an application to programming of the text *A Logical Approach to Discrete Math* —"LADM"— which has been the main course text for the past six years for COMP SCI 1FC3 and recently SWFR ENG 2DM3.

Additional material will be handed out or made available electronically via the course pages.

Outline:

The main theme of this course is the *calculation* of correct programs.

We start with asking why this is needed and briefly exploring the approach in the C programming language and using the tool Frama-C for static analysis of our programs. Then we move on to the formality of it all, the foundational logics.

We do not dwell much on the elementary logics since we expect the student to have seen them in a discrete math class —we will however review what we need!

Afterwards, we consider proof rules that guide us in constructing code from specifications. To avoid having to learn a complete programming language, we use an idealised simple language with only the bare necessities — assignment, sequential composition, conditionals, and loops. Afterwards, we try out our new skills in Frama-C.

Finally, we consider the dual to imperative programming: functional programming. We present this by briefly looking at constructive mathematics in the verified setting of Agda —which essentially type checks our mathematics.

For simplicity, tool use will be kept to a minimum but its importance cannot be overstated!

• Why program construction and correctness	Chapters 1, 4, 6	≈ 1 week
• Beginning C Programming, and Static Analysis of C Programs in Frama-C	Frama-C documentation	≈ 1 weeks
• Propositional Logic	Chapters 5, 7, 8	≈ 2 week
• Predicate Logic	Chapters 11–12	≈ 1 week
• Simple Imperative Programming, Hoare Logic	Chapters 9, 10, 13	≈ 2 weeks
• Techniques for finding invariant and Frama-C again	Lecture Notes	≈ 1 weeks
• Constructive Logic and Agda	Agda-wiki	≈ 2 weeks
• Certified Functional Programming	Lecture Notes	≈ 2 weeks

(With the most relevant textbook chapters indicated — not all textbook contents will be covered in detail, and material will be interleaved heavily. Times are rough estimates.)

Exercise Sheets and Tutorials:

In most weeks, an **Exercise Sheet** will be provided, whose questions will constitute the main material for the tutorials. They are not to be handed-in but serve as preparation for the **Bi-weekly tests**.

Every week, starting January 8, there will be one-hour tutorial session. The main purpose of the tutorials is to discuss **student work** on exercise problems. Therefore, every student is expected to complete the scheduled work, i.e., exercise problems or necessary reading, **before** the corresponding tutorial session — in particular, solutions and solution attempts to the Exercises and Test Questions on the weekly exercise sheets are to be brought to the tutorial.

Grading:

All examinations in this course will be **Closed Book**. That is, no written or printed material nor a calculator nor other electronic aids.

> After notations, presentation rules, and basic definitions and proof rules have been introduced in class, **students are expected to know them at all times**.

Bi-weekly Tests: There will be **six bi-weekly tests**, so that students have an incentive to be caught-up with the class. Each test will be between 20 and 50 minutes in length; the current plan is that these will be written **on every other Monday in place of lecture**.

Quiz 1:	Monday, 16th January
Quiz 2:	Monday, 30th January
Quiz 3:	Monday, 13th Feburary
Quiz 4:	Monday, 6th March
Quiz 5:	Monday, 20th March
Quiz 6:	Monday, 3rd April

Final Exam: The **final examination** will be scheduled by the Registrar's Office in the usual way. It will be a closed book examination of 2.5 hours (150 minutes) duration and cover the material of **all** lectures, tutorials, handouts, and quizzes.

Grade Calculation: All exam grades will be percentage grades.

For every student,

- Each quiz is worth 9%.

  Some quizzes may contain **bonus questions**. All bonus marks will be added to the course grade *only for those who have passed the course otherwise.*

- Participation is worth 6% and includes **submission of a letter of introduction** for each student and is due on January 13th. More information will be provided in class.

- The remaining 40% is given to the **final exam**.

The final course grade will be converted from a percentage grade to a letter grade according to the scale of the Registrar's Office.

The instructor reserves the right to conduct any deferred quiz or final exams orally.

Course Adaptation

The instructor and university reserve the right to modify elements of the course during the term.

The university may change the dates and deadlines for any or all courses in extreme circumstances. If either type of modification becomes necessary, reasonable notice and communication with the students will be given with explanation and the opportunity to comment on changes.

It is the responsibility of the student to check their McMaster email and course websites weekly during the term and to note any changes.

Academic Ethics

You are expected to exhibit honesty and use ethical behaviour in all aspects of the learning process. Academic credentials you earn are rooted in principles of honesty and academic integrity.

Academic dishonesty is to knowingly act or fail to act in a way that results or could result in unearned academic credit or advantage. This behaviour can result in serious consequences, e.g. the grade of zero on an assignment, loss of credit with a notation on the transcript (notation reads: "Grade of F assigned for academic dishonesty"), and/or suspension or expulsion from the university.

It is your responsibility to understand what constitutes academic dishonesty. For information on the various types of academic dishonesty please refer to the Academic Integrity Policy, located at `http://www.mcmaster.ca/academicintegrity`.

The following illustrates only four forms of academic dishonesty:

1. Plagiarism, e.g. **the submission of work that is not one's own** or for which other credit has been obtained.
2. **Collaboration where individual work is expected.**

   You **are allowed**, and encouraged, to collaborate on the **exercise questions**. (The tutorials are typically not expected to cover all exercise questions.)
3. Improper collaboration in group work.
4. **Copying or using unauthorised aids in tests and examinations.**

Academic Accommodation of Students with Disabilities

Students who require academic accommodation must contact Student Accessibility Services (SAS) to make arrangements with a Program Coordinator. Academic accommodations must be arranged for each term of study. Student Accessibility Services can be contacted by phone 905-525-9140 ext. 28652 or e-mail sas@mcmaster.ca. For further information, consult McMaster University's Policy for Academic Accommodation of Students with Disabilities.

Discrimination

The Faculty of Engineering is concerned with ensuring an environment that is free of all adverse discrimination. If there is a problem that cannot be resolved by discussion among the persons concerned, individuals are reminded that they should contact the Department Chair, the Sexual Harassment Office or the Human Rights Consultant, as soon as possible.

Learning Objectives

Course Precondition

Students are expected to have achieved the following learning objectives before taking this course:

1. Basic number types, such as integers, rationals, reals and their some of their arithmetical properties
2. General principles of mathematical notation
3. General principles of calculation in mathematics
4. Basic imperative programming, including notions of typed expressions
5. Practical syntax of propositional and predicate logic with calcuational proofs —these will be reviewed!
6. Basic data structures and algorithms —these will be reviewed as needed!
7. Perform calculations and solve equations in numerical domains
8. Explain and prove basic properties using induction
9. Write and "mentally execute" simple imperative programs
10. Translate back-and-forth between predicate-logic formulae and English statements of moderate complexity or mathematical prose. —this will be reviewed!

Course Postcondition

Students are expected to achieve the following learning objectives at the end of this course:

1. Explain and prove basic graph properties and sorting algorithms.
2. Principles of (dependently) typed expressions, and the types of the operators they are using
3. Principles of calculational proofs for propositional and predicate logic
4. Big-step operational semantics of a simple imperative programming language
5. Hoare logic proof rules for a simple imperative programming language
6. Translate English specifications of program fragments into formal pre- and post-condition specifications
7. Use Hoare logic to prove partial and total correctness of simple imperative programs.
8. Annotate their programs with appropriate specifications and assertions for mechanised analysis with Frama-C.
9. Use predicate logic for procedure and datatype specification
10. *Derive* simple imperative algorithms from their formal specifications.
11. *Calculate* efficient implementations of simple functional algorithms from clearly correct but inefficient implementations.
12. Scope and limitations of automated verification, proof, and program analysis tools.
13. Have elementary familiarity with certified programming: encoding the specification of a program into the type using Agda.
14. Understand the Curry-Howard correspondence: the usefulness of making proofs program-like (CalcCheck!) and the usefulness of making program's proof-like (Agda!).

Just as one of the main goals of this course is the study of pre- and post-conditions for programs, it is reasonably expected that such a course itself comes with pre- and post-conditions ;)