Module Code	CSU44004, CS55004
Module Name	Formal Verification
ECTS Weighting[1]	5 ECTS
Semester taught	Semester 1
Module Coordinator/s	Dr Vasileios Koutavas
Module Learning Outcomes	On successful completion of this module, students will: LO1. Have gained significant knowledge of mathematical logics (propositional and first order logic) and the Floyd/Hoare logic for software specification. These logics are the basis for software verification, and are a de facto standard in the verification industry. LO2. Be able to understand the current state-of-the-art in software verification technology, its range of applicability and theoretical limitations, and the necessary tasks required to overcome these limitations where possible. LO3. Be able to use logic formulas to specify the correct behaviour of programs, including programs implementing well-known algorithms (e.g. binary search), new programs provided to them, and programs they write themselves to solve engineering problems. LO4. Learn to use the state-of-the-art in software verification tools (e.g. Microsoft Dafny) to mechanically verify software correctness in a team setting. LO5. Learn how to use pencil-and-paper mathematical proofs to manually verify software correctness. LO6. Understand the impact of software faults in different areas of engineering, such as aerospace and medical, and how this translates to financial, ethical, and human well-being.
Module Content	Specification languages and logics; axiomatic program semantics. Formal proof systems to verify software and system properties such as propositional, predicate and Hoare logic. Proofs by induction. Correctness proofs of functional and imperative programs.
Teaching and Learning Methods	Lectures, tutorials, individual assignments, projects. Lectures will be split between presentation of new material and tutorial-style problem solving. Students will have the opportunity to practice the techniques learned in the module, and improve their skills by individual assignments and group project work.

[1] TEP Glossary

Assessment Details[2]

Assessment Component	Brief Description	Learning Outcomes Addressed	% of total	Week set	Week due
Examination	2 hour written examination	LO1, LO2, LO3, LO4, LO5, LO6	65%	n/a	n/a
In-class participation	In-class participation		2%	Week 1	Week 12
Assignment 1	Mathematical Logic	LO1, LO3, LO5	4%	Week 2	Week 4
Assignment 2	Hoare Logic/Software Verification	LO1, LO2, LO3, LO4, LO5, LO6	5%	Week 4	Week 6
Assignment 3	Hoare Logic/Software Verification	LO1, LO2, LO3, LO4, LO5, LO6	7%	Week 6	Week 8
Project Part 1	Write a program	LO1, LO2, LO3, LO4, LO5, LO6	2%	Week 2	Week 4
Project Part 2	Give formal spec of part 1	LO1, LO2, LO3, LO4, LO5, LO6	5%	Week 4	Week 8
Project Part 3	Verify spec of part 2	LO1, LO2, LO3, LO4, LO5, LO6	10%	Week 8	Week 11

Reassessment Details

Examination (2 hours, 100%)

Contact Hours and Indicative Student Workload

Contact Hours (scheduled hours per student over full module), broken down by:	**33 hours**
lecture	33 hours
laboratory	0 hours
tutorial or seminar	0 hours
other	0 hours
Independent study (outside scheduled contact hours), broken down by:	**72 hours**
preparation for classes and review of material (including preparation for examination, if applicable)	36 hours
completion of assessments (including examination, if applicable)	36 hours
Total Hours	**105 hours**

Recommended Reading List

Main textbook:
Logic in Computer Science: Modelling and Reasoning about Systems, 2nd Edition by Michael Huth and Mark Ryan

Lecture notes and handouts.

Module Pre-requisites

Prerequisite modules: none

Other/alternative non-module prerequisites: math, programming

Module Co-requisites

Module Website

Last Update

18/06/2019 by Vasileios Koutavas

[2] TEP Guidelines on Workload and Assessment