MODULE 2	FOUNDATIONS OF COMPUTING	
CREDIT POINTS	10	
STATUS	Core	
ASSESSMENT	Continuous Assessment	60%
	Examination	40%
TOTAL CONTACT HOURS: 96		
Lecture: 72	Practical:	
Tutorial: 24	Other:	
TOTAL STUDENT EFFORT: 200		

Aims

This module will teach you to calculate using discrete structures. These structures form the basis of every computer system. You will learn how to represent problems in a mathematical language that allows effective reasoning. You will learn how to manipulate these structures so as to generate efficient solutions to these problems.

You will also learn of the limits of the computer, those problems that the computer needs vast amounts of time to solve, and those problems that current computational models will never be able to solve. You will learn of suggested techniques and models that may overcome (or at least reduce) these limits such as randomisation, parallelism, quantum computing, and molecular computing.

Learning Outcomes

Upon successful completion of this module, you should be able to:

1. solve simple problems effectively using a selection of algorithmic techniques such as invariants

2. reason algebraically in a calculational style with Boolean expressions

3. use concepts and notations of discrete maths to formulate simple models and reason about them by calculation

4. apply calculational techniques effectively to a selection of problem domains in computing

5. explain using vivid examples the underlying ideas of computation, and outline how they can be modelled mathematically

6. implement programs to solve certain mathematical problems

Indicative Content

Topic	Description
Introduction and motivation	Why study foundations? The power of mathematical thinking. How mathematics 'works'. Simple motivational case studies.
Algorithmics	Informal but rigorous exploration of notions of algorithm, computational process, program, programming language, programming, specification, correctness, efficiency, unsolvability, intractability, classes P and NP, parallelism, randomization, quantum computing, molecular computing.
Algorithmic problem solving	Invariants, exploiting symmetry, case studies (e.g. knights and knaves, river crossing, games).
Review of algebra	Expressions, constants, variables, operators, parenthesisation, expression 'trees', operator precedence, order of association, datatypes. 'Laws' of Algebra. Exercises in algebraic manipulation. Calculational format. Importance of explanatory hints.
Predicate calculus	Boolean operators, laws, quantification. Extensive exercises in algebraic calculation. Formulating English statements as Boolean expressions.
Discrete mathematics: modelling and calculation	Sets, relations, functions, sequences, bags, numbers, graphs. Algebraic laws. Formulating things using these concepts and notations. Reasoning (by calculation) about 'models'. Comparing alternative 'models' by calculation. Structuring large formulas.
Implementation of mathematics	Elements of functional programming. Expressions. Definitions. Evaluation. Lazy versus eager evaluation. Pattern matching.