Module Specification

Module Summary Information

1	**Module Title**	Discrete Mathematics and Declarative Programming
2	**Module Credits**	20
3	**Module Level**	5
4	**Module Code**	CMP5344

5	**Module Overview**

This module embeds core discrete mathematics concepts relevant to programmers, within the context of programming. We will use sound abstract reasoning techniques to **design** and **produce** computer programs within the declarative paradigm. Finally, these solutions will be **tested** and **evaluated** against their functional requirements using a number of **testing** and **verification** techniques.

Rationale

Relevance to you (the student)

Discrete Mathematics is one of the core foundational topics that underpin the field of Computer Science by studying mathematical topics such as logic and set theory you are provided with the opportunity to further develop your skills in abstract reasoning. These 'thinking' skills are vital to a sound understanding of the mathematical/scientific principles underpinning the subject of Computer Science and are key to providing you with the kind of knowledge and skills that can result in major breakthroughs within the field.

Declarative Programming is a so-called 'Programming paradigm' it encompasses the sub-fields of Logic and Functional Programming; both of these sub-fields contain ideas that are vital to the development of a well-rounded Computer Scientist and are useful in developing a wider knowledgebase in the toolbox of professional programmers. By exposing you to different paradigms it should become clear that paradigms are far more important than individual programming languages in how they can help (or hinder) your approach to solving a problem.

Alignment with Programme philosophy

The module is intended to further progress you (the student) in your journey towards developing a sound theoretical and practical base in Computer Science and Software Development.

Our Computer Science programme contains three core threads that are woven throughout:

1. Mathematical and Analytical skills
2. Programming skills
3. Contemporary developments in the Computing Industry

Whilst it is obvious how points **1** and **2** are relevant to this module (it is of course based on Mathematics and Programming), the topics covered in this module are also vital in developing your

understanding of the principles behind many major recent (and of course historical) developments in the Computing industry.

An example:
Functional Programming techniques have been found to be extremely useful in large scale distributed data processing tasks within data science/analytics; for instance, Google's own internal web indexing systems have seen the application of functional programming concepts to index a sizable subset of the web in a relatively small amount of time. In fact, the 'Hadoop' ecosystem now common in Data Science circles is based on Google's MapReduce framework, a technology that combines distributed computing with functional programming techniques to allow for thread-safe computations over large-scale data-sets (VERY Big Data). Thus Functional Programming techniques naturally lend themselves to producing concurrent, parallel and even distributed computation systems, which is seeing increasing importance in today's world of multi-core machines and 'Big data' analytics.

Alignment with Learning and Teaching strategy
The learning and teaching approach taken for this module is a based on a 'flipped' approach to learning, therefore you will be expected to engage in set learning tasks each week in addition to (prior to and after) the formal class contact time. In the interests of motivating your study of the topics covered within this module we will embed the core mathematical and other theoretical concepts within your study through the use of practical programming tasks and puzzles for you to complete. Your coursework is also directly aligned with the taught topics of each week meaning that there are direct links between the knowledge skills you should learn in the lab exercises and how to apply them to your coursework (in fact some weeks may provide hints/examples on how these topics may be applied to your coursework solutions).

6	Indicative Content

- Introducing Declarative Programming
- Logic and Set Theory
- Graph theory (Graphs and Trees)
- Equational Reasoning
- Recursion
- Higher Order Programming
- Abstract Data Types
- Testing and Verification
- Logic Programming

7	Module Learning Outcomes On successful completion of the module, students will be able to:	
	1	Design well-founded solutions to problems through the use of discrete mathematics, algorithm design/analysis and Declarative programming techniques.
	2	Implement programming solutions within the declarative paradigm, which adequately meet a given specification.
	3	Apply appropriate methods, both manually and through the use of software tools, to test and verify program code.
	4	Demonstrate the ability to present a team-produced Software solution to stakeholders
	5	Discuss key concepts which underpin the Declarative Programming paradigm

8	Module Assessment			
Learning Outcome				
		Coursework	**Exam**	**In-Person**
1-4				X
5			X	

9	Breakdown Learning and Teaching Activities	
Learning Activities		**Hours**
Scheduled Learning (SL) includes lectures, practical classes and workshops, peer group learning, Graduate+, as specified in timetable		48
Directed Learning (DL) includes placements, work-based learning, external visits, on-line activity, Graduate+, peer learning, as directed on VLE		36
Private Study (PS) includes preparation for exams		116
Total Study Hours:		200