

## **MODULE: INTRODUCTION TO FORMAL DESIGN (course code BSCH-IFD)**

**Griffith College Dublin – Computing Science**

**Fall semester**

**This module is intended for junior and senior level students who are majoring in this field.**

### **Aims**

This module aims to understand and apply those branches of logic necessary to develop correct programs; to develop an understanding of the mathematics required to develop programs from initial specifications through to implementation by using formal techniques; how to prove the correctness of a given program based on its pre and post conditions; apply a formal semantic model to the derivation of programs; methods necessary to derive correct programs given a formal specification in terms of pre/post conditions.

### **Learning Outcomes**

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

1. use assertions to prove the axiomatic semantics of a simple programming language
2. use pre/post conditions to prove the correctness of simple programs
3. prove the correctness of programs using loops, loops within loops and sequential loops
4. derive programs from initial specifications using different methods of construction based on axiomatic semantics

### **Indicative Content**

<b>Topic</b>	<b>Description</b>
<b>Introduction</b>	Explain the need for formal methods in the construction of programs; Examples of programs with bugs; Writing assertions over sequences; The use of predicates to describe states in programs.
<b>Axiomatic semantics of a guarded command language</b>	Predicates as assertions in programming languages; The role of assertions in the execution of programs. The semantics $\{P\} S \{Q\}$ ; Definition of skip; assignment; if .. fi; do .. od; Definition of an invariant and its use in proving the correctness of loops; Proving correctness using definitions; Proving correctness of complete programs using assertions

	and axiomatic semantics;
<b>Formal derivation of programs</b>	<p>Writing specifications of problems using pre conditions and post conditions;</p> <p>Formally deriving programs from initial specifications to complete program code;</p> <p>Method 1: Problems of the form</p> $\{P\} \text{ do } b \rightarrow S1 \{P\} \text{ od } \{P \wedge Q\}$ <p>Method 2: Replacing a constant with a variable;</p> <p>Problem domains involving loops within loops;</p> <p>Method 3: Strengthening an invariant;</p> <p>Invariant diagrams;</p> <p>Applying formal approach to searching and sorting;</p> <p>Searching for optimal solutions – <math>O(\log N)</math> and <math>O(N)</math> solutions to computational problems.</p>

### Assessment Methods

To assess this module we will use three different modes: an open book exam, a problem sheet and a formal closed book examination. The first assessment will consist of an open book exam that will be taken after completing the first part of the course that covers the basic mathematical principles. This will last for one hour and will have a weighting of 15%. The second assessment mode will consist of a problem sheet that students are required to solve in their own time. Again it will have a weighting of 15%. The examination paper will consist of five questions. Question one will be compulsory and will carry a mark of 40. The remaining questions will each carry a mark of 20. Students will be expected to complete 3 of these questions.