MODULE 17	INTRODUCTION TO FORMAL DESIGN	
CREDIT POINTS	7.5	
STATUS	Core	
ASSESSMENT	Continuous Assessment	50%
	Examination	50%
TOTAL CONTACT HOURS: 60		
Lecture: 24	Practical: 36	
Tutorial:	Other:	
TOTAL STUDENT EFFORT: 150		

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

Topic	Description
Introduction	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.
Axiomatic semantics of a guarded command language	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;
Formal derivation of programs	Writing specifications of problems using pre conditions and post conditions; Formally deriving programs from initial specifications to complete program code; Method 1: Problems of the form $\{P\}\ do\ \ b \rightarrow\ \ S1\ \{P\} od\ \{P \wedge Q\}$ Method 2: Replacing a constant with a variable; Problem domains involving loops within loops; Method 3: Strengthening an invariant; Invariant diagrams; Applying formal approach to searching and sorting;

	Searching for optimal solutions – $O(\log N)$ and $O(N)$ solutions to computational problems.