Module: Programming 181

Module name:	Programming 181
Code:	PRG181
NQF level:	5
Type:	Core – Bachelor of Computing (all streams)
Contact time:	72 hours
Structured time:	10 hours
Self-directed time:	78 hours
Notional hours:	160 hours
Credits:	16
Prerequisites:	None

Purpose

The purpose of this module is to understand how software has helped people solve problems. The student will learn several general concepts that will allow them to formulate and understanding of a problem and develop an algorithm to support the solution. They will be introduced to arithmetic used in programming, sequences, selection and iteration control structures with built in data types.

Outcomes

Upon successful completion of this module, the student will be able to:

- Demonstrate an informed understanding of problem solving concepts, binary logic, flowcharts, pseudo code, built in datatypes and algorithms found within the software domain.
- Select and apply standard problem solving techniques within the software discipline, and to plan and manage an implementation process applied to problem solving.
- Identify, evaluate and solve defined arithmetic problems, and to apply solutions based on relevant evidence and understanding the consequences if an algorithm is poorly designed within a computer systems.
- Operate in a range of familiar problem domains, demonstrating an understanding of different kinds of problems to be solved, their constituent parts and the relationships between these parts, and to understand how algorithms in one area impact on other areas within the same software system.

Assessment

- Continuous evaluation of work through 3 assignments.
- Continuous evaluation of work through formative tests and summative test which assesses the theoretical knowledge.
- Continuous evaluation of project work, whereby the student must evaluate and present results on given algorithmic problems.
- Final assessment through a written examination.
- The assignments or projects collectively will count 30% of your class mark.
- All tests will collectively account for 70% of your class mark.

- Your class mark contributes 30% towards your final mark for the subject, while the final assessment accounts for 70% of your final mark.

Teaching and Learning

Learning materials

Prescribed Material(EBSCO)

- *A., B. (2016)* Problem Solving and Computer Programming Using C*. New Delhi: Laxmi Publications Pvt Ltd.*
- *Karl Beecher (2017)* Computational Thinking : A Beginner's Guide to Problem-solving and Programming*. Swindon, UK: BCS, The Chartered Institute for IT.*
- *Srivastava, A. K. (2020)* A Practical Approach to Data Structure and Algorithm with Programming in C*. Oakville, ON: Arcler Press.*

Additional Reference Material

- Sprankle, M., Hubbard, J. (2011). *Problem Solving and Programming Concepts* (9th Edition). Pearson.[ISBN-13: 978-0132492645]

Learning activities

The teaching and learning activities consist of a combination of formal lectures on theoretical and practical concepts, exercises and discussions. Three mandatory assignments and two projects must be completed during the course. The experiences and progress on these practical components form the content of class discussions.

Notional learning hours

Activity	**Units**	**Contact Time**	**Structured Time**	**Self-Directed Time**
Lecture		52.0		26.0
Formative feedback		13.0		
Project	2	7.0		12.0
Assignment	3			9.0
Test	4		8.0	16.0
Exam	1		2.0	15.0
		72.0	**10.0**	**78.0**

Syllabus

- Reflect on how the creation of software has changed our lives.
- Synthesize how software has helped people, organizations, and society to solve problems.
- Describe several ways in which software has created new knowledge.
- Select the appropriate flowchart and logic tools to represent, and process program data.
- Explain the appropriateness of iterative and recursive problem solutions.
- Develop a correct program to solve problems by using an iterative process, documentation of program components.

- Describe why and how algorithms solve computational problems.
- Create algorithms to solve a computational problem.
- Explain how programs implement algorithms in terms of instruction processing, program execution, and running processes.
- Evaluate various built-in types for the chosen programming language to facilitate algorithm design.
- Create logic gates and use a truth table to illustrate the possible outcomes of a given scenario.
- Write the Pseudocode to describe the operating principle of a computer program or other algorithm.
- Basic understanding of Array and Set collections in use and should be able to demonstrate their understanding.