Rationale:

Programming, using many styles and languages, provides solutions to a wide variety of scientific, engineering and business problems. Programming is a core skill that will be used throughout a computer practitioner’s career. It is a skill acquired largely by practice and experience. Learning how to program requires a disciplined and structured approach in order to encourage good practice and to assist in the development of easily maintained systems. This module introduces candidates to the fundamental concepts of programming with the emphasis being laid on the whole of the software development process.

Candidates are free to use the programming language of their choice but it is recommended that a C type language is capable of being understood, as any examples of code given in the examination paper will be based on C.

Aims:

- To stress the importance of good design, documentation and usability
- To emphasise skills in problem solving and algorithm specification rather than just writing syntactically correct code
- To introduce a systematic approach to algorithm development which will assist in subsequent programming and system design modules
- To introduce candidates to the environment in which software is developed and to the tools that assist in this process

Objectives:

- Distinguish between systems software and application software
- Understand the phases of software development
- Be able to develop and understand algorithms
- Be able to develop code from algorithms in a 3rd generation high level programming language
- Be able to follow 3rd generation high level code and apply modifications to it
- Develop competence in the techniques of systematic problem analysis, program construction and documentation
- Gain an understanding of the basic concepts of good user-interface design
- Be able to test and document programs
- Gain an understanding of the principles of multiple module program construction
- Understand the need for compilers, interpreters, code generators
- Develop a knowledge and understanding of a range of fundamental algorithms
Prior Knowledge Expected:
None

Content:

FUNDAMENTAL CONCEPTS OF THE PROGRAMMING PROCESS

Concept of an algorithm

Development and semi-formal specification of algorithms, based on a simplified computer model

Development of code from an algorithm

PHASE-SPECIFIC ISSUES OF SOFTWARE DEVELOPMENT

Development techniques such as modular programming, defensive programming or recursion

Approaches to software build, such as evolutionary prototyping or 4GL development

Objectives and principles of testing and test-case specification

Testing and debugging strategies including dry-running, white-box and black-box

Styles of software documentation, such as for users or support personnel

Content of software documentation such as GUI descriptions or maintenance details

INTRODUCTION TO PROGRAMMING CONCEPTS

Types: numeric and non-numeric, elementary and derived, subtypes, and expressions such as assignments, input/output

Control structures: selection and iteration

Subprograms: procedures and functions

Data structures: Arrays (1- and 2-dimensions), implementation of queues, stacks and
lists.

Concept of data abstraction

Sorting and searching algorithms: comparative effectiveness with respect to computation and storage of scanning versus indexing methods

FILES: SEQUENTIAL, INDEX-SEQUENTIAL AND RANDOM ACCESS

Comparative effectiveness of storage and retrieval for applications such as batch processing or on-line query or both

INTRODUCTION TO CONCEPT OF USER-INTERFACE DESIGN

User requirements and characteristics of user interfaces; principles and techniques of dialogue control, navigation and selection

ROLE AND NEED FOR SYSTEM SOFTWARE

System software and its relation to application software

CASE STUDIES IN PROBLEM SOLVING/ALGORITHM ANALYSIS