

AM SYLLABUS (2008-2010)

COMPUTING

AM 07

SYLLABUS

1. INTRODUCTION

This MATSEC Advanced level Computing Syllabus has been prepared and compiled in line with previous syllabi, latest Computing related developments and space for future syllabi to add and enhance the contents. Furthermore, in line with the objectives clearly set in the previous version of this syllabus, the programming language that is to be used throughout this syllabus is to be “Java”. This language will displace the use of the “Pascal” programming language in every aspect of this syllabus wherever the use of a programming language is required, including the implementation of the project.

It is also stressed that all theoretical treatment of topics should be adequately accompanied by practical (real-world) examples when and wherever applicable.

The nature of the project and what it aims to exercise has also been amended in this syllabus. The project now carries 20% of the global score for this subject and is primarily intended to consolidate and assess the freshly acquired programming and logical-analytical skills of candidates, rather than the abstract-analytical, design, project management skills, and software engineering rationale in general, which can be better assimilated in later stages of the educational process, when it is felt that candidates’ thought processes have reached higher levels of maturity.

Candidates are expected to have followed the Computer Studies stream at secondary level and have progressed on to the SEC level Computer Studies thereby acquiring sound background knowledge of the history of Computing, current and future trends, as well as the sector-specific and social applications of Information Technology.

This syllabus is ideal for those students who wish to deepen their understanding of Computing possibly with an eye to pursuing any undergraduate degree programme with a fundamental treatment of the technical aspects of Computing.

This document is organized as follows. The next section briefly underlines the contents of the examination itself while syllabus details are described in detail in Section 3. Section 4 lists recommended texts and reference books that Computing educators can make use of to assist their students. Finally, detailed information about the practical project is expanded in Section 5.

2. EXAMINATION

The examination shall consist of three parts, namely, two written papers each of three hours duration and a project. The overall grade will be based on an overall aggregate score, and students must obtain a minimum mark in each part of the examination, to be established by the Markers’ panel.

Paper I (100 marks) shall consist of twenty short compulsory questions from any part of the syllabus and that require short to the point answers worth 5 marks. This paper carries 40% of the final (total) examination score.

Paper II (100 marks) shall consist of eight long questions of which candidates are expected to answer five. Each question carries 20 marks and students are expected to understand precisely what is being asked and demonstrate understanding by answering in some depth. This paper carries 40% of the final (total) examination score.

Paper III (100 marks) is a project that is expected to take up about three months of the candidates’ study programme in order to clearly test and demonstrate the range of logical and programming skills they have acquired and possess. The project carries 20% of the final (total) examination score.

Note regarding use of calculators:

Calculators may **NOT** be used in any part of this examination.

3. SYLLABUS

Module 1: Digital Logic

Objectives

Students should be able to

- understand the basics behind binary logic
- make use, understand and draw truth tables for logic expressions
- draw logic circuits from Boolean expressions
- apply fundamental Boolean algebra rules and/or Karnaugh maps to simplify simple Boolean expressions
- understand the use of basic logic theorems to build practical and fundamental logic circuitry

<i>Data Representations</i>	Decimal, Binary and Hexadecimal number systems Converting numbers from one number system to another The use of sign and magnitude and two's complement method to represent positive and negative numbers The interpretation of different numerical representation formats: <ul style="list-style-type: none">- unsigned whole numbers- unsigned numbers with fractions- signed integer fixed point representation- signed fractional fixed point representation- signed floating-point representation The range of fixed-point format representation Overflow and underflow The use of codes to define a character set including ASCII, Unicode
<i>Logic Gates</i>	Binary logic as a mathematical way of manipulating and processing binary information Basic logic operators: AND OR NOT NAND NOR XOR Logic gates, truth tables and digital circuit symbols to represent simple logic solutions Drawing of logic circuits from Boolean expressions Using NANDs and NORs to represent any logic function
<i>Boolean Algebra</i>	Basic theorems and properties of Boolean algebra Equivalence, contradictions and tautologies The following list of laws will be assumed:

1. Commutative laws.

(a) $X + Y = Y + X$; (b) $X \cdot Y = Y \cdot X$

2. Associative laws.

(a) $X + (Y + Z) = (X + Y) + Z$; (b) $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$

3. Distributive laws.

(a) $X \cdot (Y + Z) = X \cdot Y + X \cdot Z$; (b) $X + Y \cdot Z = (X + Y) \cdot (X + Z)$

4. De Morgans laws.

(a) $\overline{(X + Y)} = \overline{X} \cdot \overline{Y}$; (b) $\overline{(X \cdot Y)} = \overline{X} + \overline{Y}$

5. Laws of absorption.

(a) $X + X \cdot Y = X$; (b) $X \cdot (X + Y) = X$

6. Double complement law.

$$\overline{\overline{X}} = X$$

7. Laws of tautology.

(a) $X + X = X$; (b) $X \cdot X = X$

(c) $X + \overline{X} = 1$; (d) $X \cdot \overline{X} = 0$

(e) $X + 1 = 1$; (f) $X \cdot 1 = X$

(g) $X + 0 = X$; (h) $X \cdot 0 = 0$

The simplification of Boolean expressions using Boolean algebra rules and/or Karnaugh maps (attention should be drawn to “don’t-care” values)

The applications which should be considered when applying the above theorems should include all of, and be limited to, the following:

- Half and full adders;
- Magnitude comparators;
- BCD-to-Gray code converters;
- 7-segment display units (excluding the decimal point).

Module 2: Computer Architecture

Objectives

Students should be able to

- gain a good understanding of all the components making up the computer system
- understand the function of the different components making up the system
- have a clear understanding of how the different system components are connected together and how they work to give the required output

Content

Overview of the Organization of a Computer System

Main PC components

- CPU
- Main Memory
- I/O Subsystem

The System Bus

System bus as a means of communication between components

- Address Bus
- Data Bus
- Control Bus

Buses size consideration

System Clock

Interfacing devices to a common bus using the method of

- Decoders

Synchronous and Asynchronous Transfer

Description of memory read and write cycles

<i>Memory</i>	<p>RAM Memory chip typical organization</p> <ul style="list-style-type: none">- Data input and output lines- Address input lines- Write enable line <p>The characteristics and application of RAM type memory chips</p> <ul style="list-style-type: none">- Dynamic RAM- Static RAM <p>ROM Memory chip typical organization</p> <p>The characteristics and application of ROM type memory chips</p> <ul style="list-style-type: none">- ROM- EPROM- PROM- EEPROM <p>Memory Address Map</p> <p>Memory connection to the CPU (address decoders)</p>
<i>I/O Subsystem</i>	<p>I/O Addressing</p> <ul style="list-style-type: none">- Memory mapped vs. Isolated/Separated I/O <p>Handshaking</p> <p>Interrupts</p> <ul style="list-style-type: none">- Overview of interrupt handling- Detecting source of interrupt- Software polling vs. vectored interrupts <p>DMA</p>
<i>CPU</i>	<p>CPU model and overview of main components</p> <p>CPU's instruction set and instruction format</p> <p>The fetch, decode and execute cycle</p> <p>Control Unit</p> <p>Arithmetic Logic Unit</p> <p>The stack and its role in subroutine transfer</p>
<i>CPU registers</i>	<p>The purpose and use of special internal registers in the functioning of the CPU including</p> <ul style="list-style-type: none">- Data registers- Segment registers- Index registers- Stack registers- Control registers- Status or flag register
<i>Overview of some CPU design issues</i>	<p>RISC</p> <p>CISC</p> <p><i>(Should be limited to a descriptive overview of the fundamental concepts and a treatment of functional and applicative differences. Examples should be used to support explanation)</i></p>
<i>I/O Peripherals</i>	<p>Serial, parallel and USB ports</p> <p>Serial data transmission</p> <ul style="list-style-type: none">- Synchronous transmission- Asynchronous transmission

Module 3: Assembly Languages**Objectives**

Students should be able to

- understand the general format of assembly language instructions
- distinguish between different types of instruction groups
- distinguish between different types of instruction formats
- distinguish between various addressing modes
- understand basic assembly programs given an instruction set
- list assembler functions and tools

Content

<i>Assembly Language Instructions</i>	Instruction sets Instruction format – opcode & operand Mnemonic representation of opcode
<i>Instruction Groups</i>	<i>See Appendix A for limited instruction set</i>
<i>Instruction Formats and Addressing Modes</i>	<i>The instruction set together with relevant descriptions will be provided as part of the question's text during examination sessions. Candidates will not be expected to write complete programs in assembly, only interpretation of assembly instructions will be examined.</i>
<i>Registers and Addressing Modes</i>	The general purpose registers: AX (Accumulator), BX (Base), CX (Count) and DX (Data) as 16 bit registers with a reference to their 8 bit high order and low order bytes.
<i>Immediate addressing</i>	Immediate addressing { <i>Format example: MOV destination, operand</i> } e.g. MOV AL,03H (move value 3 hex into reg. AL)
<i>Register addressing</i>	Register addressing { <i>Format example: MOV destination, source</i> } e.g. INC AX (increment value of AX by 1)
<i>Memory addressing</i>	Memory addressing The segment registers, namely DS, SS and CS (<i>ES can be omitted</i>), The idea of absolute and relative addressing, effective address and symbolic labels.
<i>Displacement only</i>	Displacement only: { <i>Format example: MOV destination, [address]</i> } e.g. MOV AL, [0810H] (move the contents of memory location 0810H to register AL)
<i>Register Indirect</i>	Register Indirect { <i>Format example: MOV destination,[BX]</i> } e.g. MOV AL,[BX] (move in AL the contents of memory location pointed at by BX)) The Index registers, namely SI (Source Index) and DI (Destination Index)
<i>Base plus index</i>	Base plus index { <i>Format example MOV destination,[BX+SI]</i> } e.g. MOV AL,[BX+SI] (move the contents of memory location pointed at by (BX plus offset in SI))
<i>Assembler</i>	The assembly process: assembling, linking, loading, and relocation The purpose of different types of assemblers: cross assembler, macro assembler, meta assembler

Module 4: Operating Systems

Objectives

Students should be able to

- describe different operating systems and their function
- outline their interaction
- develop a basic understanding of how operating systems manage memory and files
- understand how the operating system handles input and output operations

Operating Systems

Choice of operating system depends on the type of application software to be used

- Batch
- Online
- Multi-access
- Real time
- Network
- Process Control Operations

Job Control Languages (JCL)

To discuss use, name and give examples in the use of, and compare the features offered by some typical ones.

Process Control

- Process management
- States of a process
 - Run
 - Wait
 - Suspend
- Scheduling
 - Round Robin
 - Priority
- Deadlock
 - Recovery methods
 - Methods of minimization

Memory Management

Memory maps of single and multi programming environments

- Contiguous memory partitioning
- Logical vs. physical address spaces
- Re-locatability
- Memory fragmentation and compaction
- Pages and page frames
 - Global/local page tables
 - Size considerations
 - Faults
- Memory store protection
(to prevent processes from accessing storage allocated to other jobs)

File Management

File organization

- Management of files as stored physically
- Creating and accessing files
- Allocation of storage space
- Blocks
 - Contiguous
 - Linked
 - Indexed
- Facilities for editing the contents of files

Protection of files

- Facilities against unauthorised access
 - User ID and password
 - User home directory
 - File access restrictions (system administrator)
- File attributes
- Hardware failure

Handling of I/O operations

Devices that minimise complexity of I/O operations

- Interrupt vs. polling
- Interrupt handler
- System stack
- Multiple interrupts and interrupt priorities
- Interrupt mask register

Different Types of File Representation

Basic knowledge of different types of file representation and how they are associated with different applications:

Binary, Graphic (.BMP, .JPEG, .GIF), Sound (.WAV, .MP3), Video (.AVI), Text (.TXT), Records (.DBF), Hypermedia (.HTML), Compressed files (.ZIP)

Note: It is only expected that candidates be aware of basic characteristics and areas of use for the above representations.

Module 5: Networking and Communications

Objectives

Students should be able to

- understand the basics of transmission methods in communication
- distinguish between different categories of networks
- appreciate the purpose of a protocol in communication
- describe in broad terms various international networking communication protocols
- understand the implications added by internet on everyday life
- have a broad knowledge of some general Internet related technical terms

Content

Introduction

The use of networks to combine computing and communication technology for various types of distributed applications

Overview of the OSI model: physical, link, network, transport, session, presentation and application layers

Transport Control Protocol/Internet Protocol (TCP/IP)

Point-to-point Connections

Basics of communications

- The concept of sender, medium and receiver
- simplex
- half duplex
- full duplex

Parallel and Serial transmission (synchronous and asynchronous)

Analogue vs. digital signals

Bandwidth, Bit rate and baud rate

Modems

Modulation

- amplitude
- frequency
- phase
- pulse

Demodulation

Multiplexing

- Time Division Multiplexing (TDM)
- Frequency Division Multiplexing (FDM)

Transmission media

- cabling (twisted-pair, coaxial, optical fibre)
- satellites and wireless

Noise & Distortion

<i>Computer Networks</i>	Definition and classification of a computer network <ul style="list-style-type: none">- LAN- MAN- WAN
<i>Network Topologies</i>	The benefits and drawbacks of basic network topologies <ul style="list-style-type: none">- Bus- Ring- Star- Mesh
<i>Media Access Methods</i>	CSMA/CD and Token passing (improvements for FDDI and fast Ethernet)
<i>Switching Techniques</i>	The difference between various switching techniques and their application <ul style="list-style-type: none">- Circuit-switching- Message-switching- Packet-switching Datagram = packet of data + auxiliary control data Buffering Use of switches, hubs, repeaters, bridges and routers.
<i>Error checking and Recovery</i>	Transmission errors and methods to overcome them <ul style="list-style-type: none">- Parity (single and block) checking- Cyclic Redundancy Checking (CRC) (<i>only in principle</i>)- Message acknowledgement (Implicit and Explicit)- Retransmission schemes (Stop and Wait, Go-back-N, Selective Repeat)
<i>IP Addressing</i>	IP as an addressing scheme for every machine on the Internet for successful delivery of information, its use in subnets, DNS and URL translation
<i>Internet Applications</i>	E-learning, as a complimentary way of learning: <i>Should include aspects of a/synchronous learning, blended learning, technological support, basic pedagogical issues, Virtual Learning Environments (VLEs), and both commercial and open-source teaching platforms</i> distance learning, e-commerce, virtual worlds, conferencing, research, travel, news and communication
<i>Effective Web exploitation</i>	Literature searching Topic tracking and coverage Collaboration methods Plagiarism and professional respect Net-tiquette in general
<i>Social Implications of Computing and Internet</i>	Computer crime Computer-related security issues (data, application, networking perspectives) Computers, employment and privacy International and local legislation The global communication and its effect on concept of citizenship and culture Web 2.0
<i>Integration of Internet and WWW-related Protocols and Terminology</i>	Hypertext Markup Language (HTML), hypermedia, authoring tools, ADSL, ISDN, Telnet, FTP, IMAP, POP3 <i>Note: Only familiarity in broad terms (conceptual and comparative only) is required in this section</i>

Module 6: Language Translators**Objectives**

Students should be able to

- understand the structure of a formal language
- define the syntax of a formal language using relevant tools
- appreciate the need to define the semantics of the formal language
- describe the stages of compilation
- differentiate between various types of language translators

<i>Formal languages</i>	The differences between natural and formal languages The alphabet of the language Terminal and non-terminal symbols Sentences of the language Language productions
<i>The syntax of a formal language</i>	Symbol classes (identifiers, delimiters, operators, numbers) Reserved words The use of BNF/EBNF and syntax graphs to unambiguously express the syntax of a language The use of parse trees and canonical parsing to check that a statement is syntactically correct according to a set of rules or productions The use of Reverse Polish Notation (RPN) to define arithmetical statements
<i>The semantics of a formal language</i>	The need for semantics (meaning) other than syntax (form) Context sensitivity
<i>The compilation process</i>	The stages of compilation <ul style="list-style-type: none"> - Lexical analysis <ul style="list-style-type: none"> - removal of redundant text, simple error handling - conversion of lexemes to tokens - Syntax and semantic analysis <ul style="list-style-type: none"> - parsing - symbol table - compile-time error detection and handling - Code optimisation <ul style="list-style-type: none"> - simple techniques to optimise code - Code generation <ul style="list-style-type: none"> - translation into object code, linking
<i>Language translators</i>	The differences between assemblers, compilers and interpreters Other types of compilers: macro preprocessors, cross-compilers, p-code compilers, virtual machine concepts, just-in-time compilation.

Module 7: Systems Analysis and Design**Objectives**

Students should be able to

- Understand the reasons and benefits of system development through sound analysis and design principles
- Acquire introductory knowledge pertaining to the scope and integration of software development life cycle phases in the form of very basic domain research, requirements establishment, functional analysis, system design, implementation through coding and rendering-productive, and fundamentals of testing (concepts of test ranging and test case construction) and a feel for the importance and impact of maintenance.
- Gain a basic feel for the application of the above principles by being brought into contact with modern practical commercial software development approaches through a top-level outline and feature/scope comparison of two software development methodology standards known as Structured Systems Analysis and Design Method (SSADM) and the Unified Modelling Language (UML).

<i>Software systems</i>	What constitutes a software system? The needs of modern software system development. The importance of thinking before doing. A software development life cycle (simplistic “Waterfall”). Awareness that other life cycles exist and which development issues they target.
<i>Domain research</i>	The basic system domains (scientific, business, and control). Students should be able to gain adequate insight into the scientific or business field for which a particular system is to be developed. Techniques should include use of traditional and electronic sources, current system observations, very basic ideas regarding surveys and questionnaires, outline the nature of user/client meetings and interviews. The importance of looking at a software system as a solution to a real-world issue.
<i>Requirements establishment</i>	This should include a justification of system feasibility, but a separate treatment of system feasibility is not required. What are system requirements? Bridging the gap between the technical and the non-technical (i.e. between developer and user/client). The importance of <ul style="list-style-type: none"> – The ability to elicit system requirements by asking the right questions; – Structuring ascertaining the completeness of requirements; – Considering and presenting alternative solutions.
<i>Functional analysis</i>	Systems requirements coverage (Processing, Data, and Input-Output/GUI). Analysis as a tool for evolving requirements into internal system functions. The importance of internal functions satisfying externally observable requirements. Using data flow analysis to describe system function (Data Flow Diagrams - DFDs) as the principle analysis tool. An introduction to Entity-Event Modelling. The use of high-level flowcharts.
<i>System design</i>	Overview of Design aspects in the form of <ul style="list-style-type: none"> – Interface (GUI) – Data (internal and I/O) Introduction to the concept of “the module”. Modular system structuring. Explanation of the concepts of architectural and componential design. Top-down and bottom-up design approaches. Decision tables. I/O structure diagrams. The use of pseudo-code. Basic concepts of system deployment establishment (i.e. software components and hardware mapping issues).
<i>Implementation</i>	Programming language selection issues. The use of low-level flowcharts. Documentation levels to support system usage and coding (user guides, design assumptions and decision justification, module descriptions, and in-code comments). The importance of documentation-code matching. Input validation issues. Cross-over techniques (i.e. from existing to proposed), to briefly outline the following: <ul style="list-style-type: none"> – Direct (cut-off) – Parallel running – Phased transition (incremental)
<i>Fundamentals of testing</i>	Concepts and differences of Alpha and Beta testing. I/O testing in the form of validation of inputs and outputs through the creation of test-data within testing ranges. Selecting a testing strategy based on the nature and requirements of the system. Outline and comparison of Black and white box testing concepts.
<i>Maintenance</i>	The commercial importance of, and the negative impact (on overall quality, time and cost) it would have if not adequately considered. Outline of the three main categories of maintenance: <ul style="list-style-type: none"> – Adaptive – Corrective – Perfective
<i>Standard methodologies</i>	A top-level overview of both SSADM and UML as a framework within which software development can take place. An explanation of the scope, intent and application of each methodology. Brief list of the various features/tools that each methodology offers. A mention and brief outline of any variations arising from these two main-stream methodologies. Outline software (commercial or open source) that supports these two methodologies and/or their variants.

Module 8: Introduction to Data Structures and High Level Language Programming

Objectives

Students should be able to

- Identify and describe different programming paradigms including imperative, declarative, functional and object-oriented
- Gain a good understanding of the object oriented and imperative paradigms
- Have a good understanding of the fundamental concepts of object oriented programming including objects and classes, data encapsulation, inheritance and polymorphism
- Gain a good knowledge of the notions of class, object, attribute and operation
- Have a good knowledge of the different data types available
- Identify and have a sound understanding of the relevant programming constructs targeted at problem solving
- Select and appropriately apply standard algorithms for sorting and searching
- Know how to make use of files as a permanent type of storage mechanism

Content

High Level Languages

Introduction to programming paradigms

- characteristics of each programming paradigm including: imperative, declarative, functional, object-oriented and event-driven programming
- domain relevance of the above mentioned programming paradigms

Comparison between Object-oriented and Imperative Programming

- the need for a programming paradigm which models the real world in terms of software reusability
- the limitations of imperative programming: variable assignment rather than object manipulation
- the object-oriented solution: the use of classes and objects in problem solving

Object-Oriented Programming Characteristics

- Encapsulation (through classes and objects including attributes and operations)
- Message passing (i.e. operation invocation)
- Inheritance
- Information hiding
- Polymorphism

Data Types

Standard Types

- Numeric types and their ranges
- Character (char) and String types
- Boolean types
- Enumerated types
- Other classes as types

Constants

Variables

- Scope and visibility of variables

Control Structures

Conditional

Conditional structure statement

Looping Structures

Pre-tested loops
Post-tested loops
Nested loops

Methods and classes

The Java API (to be covered from point of view of usage and not content)
Argument passing by reference and by value
Class vs. Object (deserves good coverage)
Static classes
Abstract classes
Recursion

Data Structures

Exception Handling

Distinction between errors and exceptions
 Identification of “throwable classes”
 Use of “throws”
 Use of the “try-catch” block

Data structures and algorithms

Purpose and application of data structures

Stacks

LIFO structure
 Concept of Pointers
 Creating a stack (difference between static and dynamic structure)
 Push and Pop algorithms to add and delete elements from stack
 Traverse stack to display its contents

Linear Lists

Creating the structures

Linked Lists

Adding a node to the structure

Circular Lists

Deleting a node from the structure

Double Linked Lists

Traversing the structure

Queues

Binary Trees

Creating a tree
 Adding a node
 Deleting a node
 Traversing the tree using the three traversals

- Pre-Order Traversal
- In-Order Traversal
- Post-Order Traversal

Hash Tables

Notion of a hash table
 Creating and Updating a hash table
 Hash functions
 Collisions

Other Structures

Arrays – single and multi-dimensional

- Creating an array
- Filling in an array with data
- Displaying data from an array

 Vectors

Standard Algorithms

Sorting Algorithms

Space, time and complexity considerations for algorithms

- Insertion sort
- Selection sort
- Bubble sort
- Quick sort
- Merge sort

Use of the big-“O” notation to compare the above algorithms according to complexity criteria

Searching Algorithms

Linear Search
 Binary Search

Files and File Access

This section is to be discussed solely from an algorithmic point of view and should exclude any OS-dependent discussion.

Files

Text and Random files

- Text files – Creating, adding data and displaying contents

 Logical file organization

- Serial
- Sequential
- Direct
- Indexed

File Operations

- Creating a file
- Writing to a file
- Reading from a file
- Updating a file (inserting and deleting)
- Merging files

Module 9: Databases

Objectives

Students should be able to

- understand the basic structure, function and importance of database management systems (DBMS)
- be able to compare different database models
- appreciate the importance of relational databases over traditional file systems
- understand the logical structure and design of a relational database
- describe data models diagrammatically using entity-relationship (E-R) diagrams
- normalise a relational database up to third normal form
- apply methods and tools for database design by using currently available database packages
- understand the purpose of a query language and be able to interpret simple SQL commands

Content

Database Management Systems

The structure and functions of database management systems (DBMS) including:

- data dictionary
- file manager
- data manipulation language (DML)
- data description language (DDL)
- query language
- security

The responsibilities of a database administrator

Database Models

Comparison of flat files, hierarchical, network and relational database models, object-oriented database models.

Databases vs. Traditional File Systems

The advantages of databases over traditional file systems including: improved data consistency and portability, control over data redundancy, greater security

The disadvantages of databases over traditional file systems including: greater complexity and cost, vulnerability to system failure and unauthorised access, larger size

Relational Databases

The nature and logical structure of a relational database as a set of tables linked together using common fields.

The purpose of primary, secondary and foreign keys, attributes (field), tuples (record).

Use a short notation to represent a relational table in which the name of the table written in capitals is followed by a list of all the attributes in brackets, with the primary key underlined.

e.g. STUDENT (studid, name, surname, dob, address)

Entity-Relationship Modelling

The use of Entity-Relationship (E-R) Models to give a graphical description of the relationship between entities including:

- one-to-one,
- one-to-many and
- many-to-many relationships

The standard "Crow's Foot" notation is to be used to model and explain the above concepts.

Normalisation

The importance of normalisation to avoid unnecessary redundancy
Normalise a simple relational database up to third normal form.

<i>Database Applications</i>	The purpose and use of commercial and top-end database packages, web-based database applications Develop a simple relational database using fourth generation applications such as Microsoft Access or Delphi
<i>Structured Query Language (SQL)</i>	Understand the purpose and use of SQL commands to manipulate data including: SELECT, FROM, WHERE, ORDER BY, HAVING, GROUP BY, JOIN <i>Candidates will NOT be expected to write segments of SQL, only interpretation of SQL instructions will be examined.</i>

APPENDIX A (TO MODULE 3): ASSEMBLY LANGUAGES

Limited instruction set to be used

<i>Data Transfer instructions</i>	MOV	Moves byte or word to register or memory	
	PUSH	Push a word on stack	
	POP	Pop a word from stack	
<i>Logical Instructions</i>	NOT	Logical not (1's complement)	
	AND	Logical and	
	OR	Logical or	
	XOR	Logical exclusive-or	
<i>Arithmetic Instruction</i>	ADD , ADC	Add and Add with carry	
	SUB, SBB	Subtract and Subtract with borrow	
	INC	Increment	
	DEC	Decrement	
	CMP	Compare	
<i>Transfer Instructions</i>	JMP	Unconditional Jump	
	JE	Jump on Equal	
	JNE	Jump on Not Equal	
	JL	Jump if Less	
	JLE	Jump if less or equal	
	JG	Jump if Greater	
	JGE	Jump if Greater or Equal	
	JC, JNC	Jump on carry or Jump on No Carry	
	CALL	Call Subroutine	
	RET	Return from subroutine	
	<i>Flag Manipulation</i>	CLC	Clear Carry
		STC	Set Carry
	<i>Shift and Rotate</i>	SHL, SHR	Logical Shift Left or Right
RCL, RCR		Rotate through Carry Left or Right	

APPENDIX B: LIST OF ACRONYMS

ADSL	- Asymmetric Digital Subscriber Line
ASCII	- American Standard Code for Information Interchange
ATM	- Asynchronous Transfer Mode
BNF	- Backus Naur Form
CISC	- Complex Instruction Set Computer
CSMA/CD	- Carrier Sense Multiple Access / Collision Detect
DMA	- Direct Memory Access
DTP	- Desktop Publishing
EBNF	- Extended Backus Naur Form
ROM	- Read Only Memory
EEPROM	- Electrically Erasable Programmable ROM
EPROM	- Erasable Programmable ROM
FDI	- Fiber Distributed Data Interface
FTP	- File Transfer Protocol
HDSL	- High bit-rate Digital Subscriber Line
IMAP	- Internet Message Access Protocol
ISDN	- Integrated Services Digital Network
LAN	- Local Area Network
LIFO	- List In First Out
MAN	- Metropolitan Area Network
OSI	- Open Systems Interconnection
POP	- Post Office Protocol
PROM	- Programmable ROM
RISC	- Reduced Instruction Set Computers
SMTP	- Simple Mail Transfer Protocol
USB	- Universal Serial Bus
WAN	- Wide Area Network

4. RECOMMENDED TEXTS

(Students are urged to look for the latest editions, ISBNs will therefore vary accordingly)

4.1 Student basic text book

Heathcote, P.M., Langfield S., *A-Level Computing*, Payne-Gallway Publishers.

4.2 Other recommended text books

Brookshear, J.G., *Computer Science – An Overview*, Addison Wesley.

David, J.B., Kolling, M., *Objects First with Java*, Prentice Hall.

Wu, C.T., *An Introduction to Object-Oriented Programming with Java*, McGraw-Hill.

The use of the Internet, in the form of on-line documentation and reference sources, is strongly recommended.

4.3 Recommended reference books

<i>Computer Architecture and Assembly</i>	Abel, P., <i>IBM PC Assembly Language and Programming</i> . Prentice Hall. Kleitz, W., <i>Digital and Microprocessor Fundamentals</i> . Prentice Hall. Uffenbeck, J., <i>The 80x86 Family: Design, Programming, and Interfacing</i> . Williams, R., <i>Computer Systems Architecture</i> .
<i>Data Structures and Algorithms</i>	Carrano F. M., Prichard J. J., <i>Data Abstraction and Problem Solving with C++: Walls and Mirrors</i> . Addison Wesley.
<i>Databases and SQL</i>	Whitehorn M., Marklyn B., <i>Inside Relational Databases</i> . Springer-Verlag UK. Taylor, A. G., <i>SQL For Dummies</i> . John Wiley & Sons Inc.
<i>Digital Logic</i>	Morris, M., <i>Digital Design</i> . Prentice Hall.
<i>Networking and Communications</i>	Hodson, P., <i>Local Area Networks</i> . Continuum. Stallings, W., <i>Data and Computer Communications</i> . Halsall, F., <i>Computer Networking and the Internet</i> .
<i>Operating Systems</i>	Ritchie, C., <i>Operating Systems. Incorporating Unix and Windows</i> . Continuum. Silberschatz, A., et al., <i>Operating System Concepts</i> Tanenbaum, A.S., Woodhull, A.S., <i>Operating Systems Design and Implementation</i> , Prentice Hall Software Series.
<i>Project Management</i>	Heathcote, P.M., <i>Tackling Computer Projects</i> . Payne-Gallway Publishers.
<i>Systems Analysis and Design</i>	Kendall, J. E., Kendall E. K., <i>Systems Analysis and Design</i> . Prentice Hall. Lejk, M., Deeks, D., <i>Systems Analysis Techniques</i> , Addison Wesley.
<i>Java</i>	Deitel, P.J., Deitel, H.M., <i>Java, How to Program</i> , Prentice Hall. Schildt, H., <i>Java: A Beginner's Guide</i> , Osborne McGraw-Hill. Schildt, H., <i>Java: J2SE (Osborne Complete Reference S.)</i> , McGraw-Hill.
<i>Other</i>	British Computer Society, <i>Glossary of Computer Terms</i> , Addison-Wesley.

5. FURTHER INFORMATION REGARDING THE PROJECT

5.1 Rationale

The project is intended to be an extended exercise requiring about three months of effort, typically conducted in the second year of study. It should demonstrate a student's mastery of:

- a) The syntax and semantics of the Java programming language. The mastery of control constructs within Java as well as the application of algorithmic logic to the resolution of real-world issues. *(It should be made clear and stressed that Java is the language that must be used for the implementation of this project).*
- b) The identification of a problem domain, some basic analytical thought towards the function and design of suitable data structures and algorithms. *(Students are encouraged to identify more than one real-life application or original project that will later have to be discussed with their supervisor).*
- c) Fundamental testing procedures and the choice of test data to demonstrate the functional behaviour of a system.
- d) Documenting a system both from a technical as well as a user perspective.

Emphasis should be directed at structured and efficient programming techniques rather than on cosmetic aspects. Originality in the selection of the problem and creative solutions will be rewarded. Typical projects should put into practice concepts and techniques covered by the syllabus.

Technical documentation presented should highlight major design decisions of data structures and algorithms. Clear, concise and correct use of English is expected.

5.2 Deadlines

- a) Schools to submit Projects' Assessment First Friday of April
- b) Private Candidates Project submission First Friday of March

5.3 Procedure for Assessment of Projects

Candidates presented by Schools. Assessment of each candidate's performance in the project will be school-based and is subject to moderation by the Markers' Panel. Tutors will submit their mark, through the Head of School, to the MATSEC Support Unit, University of Malta. The school should make the project reports available to the Markers' Panel for the purpose of carrying out the moderation exercise.

Private candidates. The project reports prepared by private candidates will be assessed directly by the Markers' Panel. Such project reports should be made available at the MATSEC Support Unit, University of Malta for assessment. A percentage of these candidates will be asked to give a full presentation of their project during a personal interview with members of the Markers' panel.

In all cases. The project should include a statement certifying that the substance of the project and the report are the candidate's own work, signed by both the tutor and the candidate. The project reports will be returned to schools and to private candidates following the publication of the examination results.

Transition Clause applicable for the Years 2008 and 2009 only

This Transition Clause applies to candidates who sat for the Advanced Level Computing Examination up to the year 2007 and are re-sitting the subject.

1. Candidates who have sat for the Advanced Level Examination prior to or in the year 2007 and are re-sitting the subject may carry forward the project mark from a previous session. The project mark will be given a weighting of 20% in the examination of the years 2008 and 2009. These candidates are required to fill in a form indicating their request.
2. If candidates who have sat for the Advanced Level Examination prior to or in the year 2007 and are re-sitting the subject opt to re-submit the project they can do so by:

EITHER

- (a) presenting a new project in JAVA according to the new criteria described in this syllabus;

OR

- (b) presenting an improved version of their original project in one of the programming languages listed in the 2005-7 syllabus. These projects will be marked according to the criteria set in the 2005-7 syllabus. The weighting of the project will be 20% of the total marks of the examination in conformity with the new syllabus. Candidates who wish to take this option are to send their request in writing to the MATSEC Support Unit by the end of October of the year prior to the examination session.

5.4 Project Report

The following points should be considered by candidates when presenting their project report and other relevant material:

- a) Any CDs submitted must be clearly labelled (with the candidate's name, project title in brief, exam session date). The CD jacket must contain a clear indication of their contents how they are to be run. Only work on once-recordable CDs will be accepted. Any CDs handed in must be finalised (i.e. no open multiple recording sessions are allowed). The use of floppy discs or re-writable CDs is NOT allowed.
- b) Documentation should be presented in a neat and well organised manner.
- c) The exact aims and objectives of the project should be stated and any deviation from the approved project should be adequately justified.
- d) A clear basic project plan must be thought out and presented.
- e) A clear table of contents should be provided towards the beginning of the report.
- f) All sections within the documentation should carry clear and meaningful headings.
- g) Any diagrams should be captioned and duly referenced in the text of the report itself.
- h) Background material on the project should be included as opening material in the report.
- i) The candidate should ensure that the documentation flow allows the reader to understand and use the project.
- j) The chosen strategy for testing should be described and justified, and test data used together with test runs suitably recorded and presented.
- k) The design of the user interface should be briefly described and justified.
- l) The overall system structure should be made clear by including suitable diagrams as and whenever deemed necessary.
- m) Any techniques and tools used should be clearly defined at some point prior to their use.
- n) The use of any third party software should be justified and its use in relation to the candidate's work explained.
- o) Annotated listing of any software produced should be provided.

- p) Any unsolved issues, errors or restrictions from the original specifications should be indicated with explanations and suitable comments.
- q) Choices taken and alternatives discarded while designing should be adequately justified.
- r) A critical evaluation of the overall success of the project should be made.
- s) Ideas for possible enhancements or more general models for the problem should be discussed.
- t) The original project plan should be compared with the actual history of the project.
- u) The final report must be soft bound.
- v) The report format should adhere to the following guidelines:

Paper Size	A4
Printing	One side of the paper only
Line Spacing	1.5
Font Size	12 (some sections in 10pt OK)
Font Type	Arial
Top, Bottom, Left Margins	3cm
Right Margin	2 cm
Page Numbering	Arabic numerals, in page footer
Page identification	Candidate name, project title, month & year, in page header
Maximum Length	Approx. 10K words

For the sake of understandability, essential information, subsidiary or detailed technical material should be included in an appendix (recommended not to exceed half the size of the project report).

5.5 Grading Scheme

The overall grade will be based on an overall aggregate score, and students must obtain a minimum mark in each paper to be established by the Markers' Panel.

Candidates re-sitting their examination at the next possible occasion may either submit a new project or transfer the project mark from the previous session.

5.6 Project Marking Scheme (guidelines for project assessors)

The award of marks will be based on the following assessment criteria, each of which carries the following weighting.

(a) Problem domain (25 marks)

Suggested subdivision of marks:

Problem definition clarity (5 marks)

The clarity and precision of how the problem was highlighted and presented.

Problem scoping and feature specification (5 marks)

How well the problem was delineated in terms of system boundaries and its expected features clearly stated.

Level of academic challenge (10 marks)

The level of complexity of the chosen task and how adequately this was achieved.

Justification of program implementation method (5 marks)

How well did the candidate justify his/her choice of solution in algorithmic terms (i.e. the soundness of program logic)?

(b) Design and implementation (35 marks)

Suggested subdivision of marks:

Program structural design (20 marks)

How well was the static structure of the program designed and presented? This should include any relevant flow-chart models and their correct and effective usage.

Data structure and variable usage (5 marks)

How sensible are the data structures constructed in relation to the needs of functions within the program?

Algorithmic/Logical reasoning (10 marks)

An assessment of the logical decisions taken by the candidate and embodied by the implemented program.

(c) Testing/Evaluation (20 marks)

Suggested subdivision of marks :

Testing procedures, values and results (5 marks)

The rationale behind the choice of tests, presentation of the values supplied and the results obtained.

System screen-shots (5 marks)

The presence of relevant and clear screen-shots.

Traceability (5 marks)

Being able to link an initially-stated feature to an actual implemented function.

Solution evaluation (5 marks)

A critical appraisal of the presented program.

(d) Overall project quality (20 marks)

Suggested subdivision of marks:

Satisfaction of requirements (5 marks)

Were all the initially-stated features of the proposed program actually and successfully implemented?

Solution completeness (5 marks)

Does the presented program present a complete solution to the problem/issue that it is meant to solve/benefit? (i.e. relevance of solution to domain).

Quality of supporting documentation (10 marks)

The overall structure, flow, clarity and relevance of the supporting documentation.

5.7 Accredited Schools

Schools presenting candidates for this examination must normally offer full-time courses in Computing and must be accredited by the Maltese education authorities.

It is the responsibility of schools presenting candidates for this examination to ensure that they are properly equipped with the appropriate hardware equipment and software packages for any project work set for the candidates. No concession for candidates lacking the right tools and equipment will be made by the MATSEC Board.

5.8 Assessors

The teachers authorised to act as assessors of the project will be appointed by the University. Any authorised assessor reserves the right to interview any candidate of his/her choice regarding the content of the project.