IM SYLLABUS (2014)

COMPUTING IM 07

SYLLABUS

Computing IM 07	(Available in September)
Syllabus	1 Paper (3 hours) + Coursework

1. INTRODUCTION

This MATSEC Intermediate 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. It is intended as a natural progression from SEC level Computer Studies and covers a reasonable and coherent portion of the MATSEC Advanced level Computing syllabus. It is aimed at eliciting roughly one third of the effort required in the MATSEC Advanced level Computing syllabus. It should be stressed that this is not a Computer Appreciation or a Computer Applications course. 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 programming assignment.

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

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, not down to the MATSEC Advanced Computing level but enough to give a basic understanding of both the theory and techniques used in computing, together with the ability to use a programming language and basic application packages in practical situations.

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 regarding the assignments is expanded in Section 5.

2. EXAMINATION

The examination shall consist of two parts, namely, one written paper of three hours duration and one school-based programming assignment moderated by the MATSEC Markers' Panel.

The written paper (80%) will be divided in two sections, namely:

a) Section A (60 marks) will consist of ten compulsory short questions requiring to the point short answers.
b) Section B (20 marks) will consist of two more in-depth questions of which the candidates will be required to answer one.

The Set Programming Assignment will be carry 20% of the total course mark.

The assignment will be set and assessed by the educators who are teaching the subject and moderated by the MATSEC Markers' Panel. This effort should be spread out over the length of the course. A list of sample topics is included in Section 5. All marks are to be submitted to the MATSEC support unit by the date stipulated by the unit.

Private candidates are to submit the assignment to the MATSEC support unit, for assessment by the Markers' panel, by the date stipulated by the MATSEC support unit. Candidates may be called for an interview regarding their work.

Note regarding use of calculators:
Calculators may **NOT** be used in any part of this examination.

3. SYLLABUS

Module 1: Basic Computing Concepts

Objectives

Students should be able to

- Understand the basic aspects of different computer systems
- Understand the need for different computer systems used for different applications
- Understand the social and ethical implications of the use of computers.

Content

Basics of computing:

Introduction to:

- Hardware
- Software
- Data & information

Identification of different Hardware and their application in the real world. Candidates should also be aware about different types/categories of software as used in the real world, for example by category:

- Process control: e.g. controlling machines in factories
- Office automation: e.g. Spreadsheet, Word processor
- Industrial & Commercial: e.g. CAD, CAM, Architectural
- Multimedia: A/V editing, Graphics Design, Authoring tools

Evolution of computer systems

Classifications of computers:

- Mainframe Computers
- Mini Computers
- Micro Computers

Generations of Computers: Candidates should be able to differentiate between generations of computers and they should provide, at least, one example of each generation. Characteristics should only be outlined.

Components of a Computer System in details:

Input and Output Hardware Devices:

Different types of input devices and their use

Storage Hardware Devices: Diff Storage Hardware devices, Serial/Sequential Storage Devices, Direct Access Storage devices.

The Von Neumann machine architecture

Definition of a program, Application software, Applications packages, System software.

The concept of the Algorithm:

- Definition of an algorithm, the importance of an algorithm to explain and breakdown a problem when designing programs.
- Introduction to different representations of an algorithm e.g. textual, diagrammatic: pseudo code, flowcharts.
- Introduction to control and data structure of an algorithm.

E-Learning:

Electronic Learning as a complimentary form of learning:

- E-learning vs. face-to-face learning style: pros and cons

- Virtual Learning Environments: Candidates should be aware that different VLE's exist e.g. WebCT, BlackBoard, Moodle, etc.

Candidates are not expected to know properties of different VLE's, however, typical content found in a VLE is expected to be known. Differences between open source and commercial source applications.

Electronic Activities:

- E-mail: e.g. send plain text messages, attaching files.
- E-Commerce: e.g. Business to consumer, purchasing on-line
- E-Business e.g. Business to Business, EFT
- Remote Access: e.g. Working remotely, logging on remotely

An overview of the above electronic activities is examined. Comparisons including advantages and disadvantages of the above when compared with traditional methods are also expected.

Social Implication of Computing

- Computer Crime
- Privacy of Data
- Data Protection Act overview, with reference to the Maltese legislation.
- Web 2.0
- Plagiarism

Data Representations

- Decimal, Binary, Hexadecimal number systems
- Converting from one base to another of the above
- Binary two's Complement & Sign and Magnitude representations, range.
- 8421 Binary Coded Decimal
- The use of fixed point number representation to represent fractions (binary only)
- ASCII & Unicode as standard codes.

Computer Language Translation

Source Code, Translation including differences between Compilers, Interpreters, Assemblers. Machine Object Code. Pseudo Machine. Description and applications of various compilers & techniques should be overviewed. Details and functioning of these translators should not be tackled.

Module 2: 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 laws of Boolean algebra and/or Karnaugh maps to simplify a Boolean expression
- Design a combinational logic circuit using a simple practical application

Content

Logic Gates

Binary logic as a mathematical way of manipulating and processing binary information
Basic logic operators: AND NOT OR NAND NOR XOR XNOR
Logic gates, truth tables and digital circuit symbols to represent simple logic solutions
The use of NAND and NOR as universal gates
Drawing of logic circuits from Boolean expressions

Boolean Algebra

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) $X \cdot 0 = 0$

Karnaugh Maps

The use of Karnaugh maps to simplify Boolean expressions.
Only Karnaugh maps using up to three variables will be considered.

Logic Circuits

Design a simplified logic circuit including the Half Adder and Full Adder from first principles.

Module 3: Computer Architecture and Assembly Language

Objectives

Students should be able to

- gain a good understanding of the main 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
- Understand some basic assembly instructions

Content

Overview of the Organization of a Computer System	Main PC components • CPU • Main Memory • I/O Subsystem • The System clock
The System Bus	System bus as a means of communication between components • Address Bus • Data Bus • Control Bus Buses size consideration and connection to various computer components
Memory	RAM The characteristics and application of RAM • Dynamic RAM • Static RAM • Cache Memory ROM The characteristics and application of ROM • ROM • EPROM • PROM • EEPROM
CPU	CPU model and overview of main components The fetch, decode and execute cycle with relevance to the Instruction Pointer, Current Instruction Register, Memory Address Register and Memory Data Register.
Registers	The purpose and use of special internal registers in the functioning of the CPU including • Data registers (Consider the four GP registers AX, BX, CX, DX) • Stack Pointer and the stack with a description of PUSH and POP • Status or flag register considering the zero and carry flag as examples.
Assembly Language Instructions	Instruction sets Instruction format – opcode & operand Mnemonic representation of opcode
Assembly Language Instructions	Only a minimal set of Instructions need to be considered: MOV, ADD, SUB, INC, CMP, JG, JL, JE, PUSH and POP
Addressing Modes	Register – Example SUB AX,BX Immediate – Example MOV AX,000FH Direct – Example MOV BX,[2000H] *Note students will not be asked to write programs in assembly language*
Assembler	The assembly process: assembling, linking, and loading, source and object codes.

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

Content

Operating Systems

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

- Batch
- Online
- Real time
- Network

The main functions of an OS:

i. Process Control

- States of a process
  - Run
  - Wait
  - Suspend
- Scheduling
  - Round Robin
  - Priority
- Deadlock: What is deadlock?

ii. Memory Management

Memory maps of single and multi programming environments

- Logical vs physical address spaces
- Memory fragmentation and compaction
- Memory store protection
  (to prevent processes from accessing storage allocated to other jobs)

iii. File Management

File organization: Creating and accessing files

Protection of files

- Facilities against unauthorised access
  - User ID and password
  - File access rights and allocated privileges
- File attributes

v. Interrupts Handling

Interrupts

- Interrupt vs. polling
- Overview of interrupt handling
- Software polling vs. vectored interrupts

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
- Appreciate the wide range Internet related technical terms and Internet applications

Content

Point-to-point Connections	Basics of communications • Simplex • Half duplex • Full duplex Analogue vs. digital signals Bandwidth and baud rate Modems Modulation • Amplitude • Frequency Demodulation Transmission media • Cabling (twisted-pair, coaxial, optical fibre) • Satellites Noise
Computer Networks	Definition and classification of a computer network • LAN • MAN • WAN
Network Topologies	The benefits and drawbacks of basic network topologies • Bus • Ring • Star • Mesh
Switching Techniques	The difference between various switching techniques and their application • Circuit-switching • Message-switching • Packet-switching
Communication Protocols	The need for standard protocols; The 7 layer OSI model and the IP model Transmission errors and methods to overcome them; Parity (single and block) checking
International Network Standards	The dotted decimal notation form of IP addressing. Internet domain naming using DNS to assign a host name to every device on the Internet corresponding to the IP address.
Integration of Internet and WWW-related Terminology	Internet service provider, web server, web client, web browser, plug-in, search engine, home page, Uniform Resource Location (URL), bookmarks, ftp, download, upload, remote login, user ID, password, Hypertext Markup Language (HTML), hypermedia, authoring tools, ISDN, ADSL, Proxy Servers, Hubs and routers *(only a broad knowledge is expected from the students)*

Module 6: High Level Language Programming and Systems Design

Objectives

Students should be able to

- Understand and appreciate the need for programming
- Differentiate between the various mainstream programming paradigms
- Identify, understand and use the basic programming and data structures and algorithms
- Manipulate data in files
- Have a good understanding of the fundamental concepts of object oriented programming including objects and classes, data encapsulation, inheritance and polymorphism
- Understand the basic principles of systems analysis and design
- Develop a practical knowledge of the accepted mainstream stages of a software development life-cycle
- Appreciate the structure and use of the fundamental development methodologies

Content

High Level Languages	Basic Knowledge of different paradigms, specifically: • Imperative • Object-Oriented • Formal *Only a comparison of non-context free and context free languages from a formal point-of-view should be tackled* Object-Oriented Programming Characteristics • Encapsulation • Inheritance • Polymorphism
Data Types	Standard Types • Numeric types and their ranges • Character (char) and String types • Boolean types • Other classes as types Constants Variables; Scope and visibility of variables
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) Class vs. Object Static classes
Data structures and algorithms	Purpose of data structures
Stacks	Basic knowledge of the concept of a stack as a data structure
Linear Lists *Other Structures*	Basic knowledge of the concept of lists and queues as data structures Arrays – single and multi-dimensional • Creating an array • Filling in an array with data • Displaying data from an array
Sorting Algorithms	Basic knowledge of sorting algorithms in general with special reference to • Insertion sort • Bubble sort

Searching Algorithms	Linear Search
Files	Text and Random files. Text files: Creating, adding data and displaying contents File Operations • Writing to a file • Reading from a file
Software systems	The importance of thinking before doing A software development life cycle (simplistic "Waterfall")
Domain research	The basic system domains (scientific, business, and control) Traditional and electronic sources, current system observations, surveys and questionnaires, user/client meetings and interviews
Requirements establishment	What are system requirements? Bridging the gap between the technical and the non-technical (i.e. between developer and user/client) Systems requirements coverage (Processing, Data, and Input-Output/GUI)
Functional analysis	Using data flow analysis to describe system function (Data Flow Diagrams - DFDs) as the principle analysis tool
System design	Overview of Design aspects in the form of • Interface • Data Modular system structuring Top-down and bottom-up design approaches The use of pseudo-code and/or flowcharts
Implementation	Programming language selection issues. User guides and in-code comments. Input validation issues. Cross-over techniques: • Direct • Parallel running • Phased transition
Fundamentals of testing	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. Comparison between black and white box testing concepts.
Maintenance	Outline of the three main categories of maintenance: • Adaptive • Corrective • Perfective
Standard methodologies	A top-level overview of both SSADM and UML as a framework within which software development can take place.

Module 7: Databases & Database Management Systems

Objectives

Candidates should be able to:

- Understand the basic structure, function and importance of database management systems (DBMS)
- Appreciate the importance of relational databases over traditional file systems
- Understand the logical structure and design of a relational database

Content

The structure and functions of database management systems (DBMS) including:	Data dictionary Data manipulation language (DML) Data description language (DDL) Query language
Database Administrator (DBA):	The responsibilities of a database administrator The role of the DBA in the design and maintenance of the database.
Database models:	The two main type of database models namely: • Relational models • Object-Oriented Model.
Databases vs. Flat File Systems	The advantages and disadvantages of databases over traditional file systems including: improved data consistency and portability, control over data redundancy, greater security, increased maintenance.
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, fields, records. Attributes & tuples. Compound Keys.
Normalising relational databases:	Candidates must appreciate that the great advantage of using Relational database is due to the normalization process which is applied on it. Candidates are ***not*** expected to normalize databases but understand the concept behind normalization to reduce duplicate data

LIST OF ACRONYMS

ADSL - Asymmetric Digital Subscriber Line

ATM - Asynchronous Transfer Mode

BNF - Backus Naur Form

DMA - Direct Memory Access

ROM - Read Only Memory

EEPROM - Electrically Erasable Programmable ROM

EPROM - Erasable Programmable ROM

FTP - File Transfer Protocol

HDSL - High bit-rate Digital Subscriber Line

ISDN - Integrated Services Digital Network

LAN - Local Area Network

MAN - Metropolitan Area Network

PROM - Programmable ROM

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's basic text book

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

4.2. Other recommended sources

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 online documentation and reference sources, is strongly recommended.

4.3. Recommended reference books

Computer Architecture and Assembly	Abel, P., *IBM PC Assembly Language and Programming*. Prentice Hall.
	Kleitz, W., *Digital and Microprocessor Fundamentals*. Prentice Hall.
	Uffenbeck, J., *The 80x86 Family: Design, Programming, and Interfacing*.
	Williams, R., *Computer Systems Architecture*.
Data Structures and Algorithms	Carrano F. M., Prichard J. J., *Data Abstraction and Problem Solving with C++: Walls and Mirrors*. Addison Wesley.
Databases and SQL	Whitehorn M., Marklyn B., *Inside Relational Databases*. Springer-Verlag UK.
	Taylor, A. G., *SQL For Dummies*. John Wiley & Sons Inc.
Digital Logic	Morris, M., *Digital Design*. Prentice Hall.
Networking and Communications	Hodson, P., *Local Area Networks*. Continuum.
	Stallings, W., *Data and Computer Communications*.
	Halsall, F., *Computer Networking and the Internet*.
Operating Systems	Ritchie, C., *Operating Systems. Incorporating Unix and Windows*. Continuum.
	Silberschatz, A., et al., *Operating System Concepts*
	Tanenbaum, A.S., Woodhull, A.S., *Operating Systems Design and Implementation*, Prentice Hall Software Series.
Project Management	Heathcote, P.M., *Tackling Computer Projects.* Payne-Gallway Publishers.
Systems Analysis and Design	Kendall, J. E., Kendall E. K., *Systems Analysis and Design*.

	Prentice Hall.
	Lejk, M., Deeks, D., *Sytems Analysis Techniques*, Addison Wesley.
Java	Deitel, P.J., Deitel, H.M., *Java, How to Program*, Prentice Hall.
	Schildt, H., *Java: A Beginner's Guide*, Osborne McGraw-Hill.
	Schildt, H., *Java: J2SE (Osborne Complete Reference S.)*, McGraw-Hill.
Other	British Computer Society, *Glossary of Computer Terms*, Addison-Wesley.

5. FURTHER INFORMATION REGARDING THE ASSIGNMENTS

5.1 Suggested Areas for Assignments

It is important, that all assignments be fully documented, with clearly defined explanations of any decisions taken and approaches adopted, together with a description of possible fields of application for the object of the assignment.

The following are some generic examples of areas to which programming assignments can be related. However, it should be noted that teachers are free to choose any sort of programming example from any field that best suits their needs and expectations.

Mathematical type of problems:
Finding statistical mean, root mean square, finding the maximum and minimum of a set of numbers, compound interest, depreciation.

Iterative types of problems:
Factorials. Simple permutations. Combinations. Problems involving repetitive tests with loops and nested loops.

Character graphics problems:
Area filling with characters. Cursor movement. Problems involving text screen handling. Use of extended character set. String manipulation problems: Abbreviations case conversion, simple string sorting. Simple string searches. Use of string arrays.

Mapping problems:
Mapping structures onto strings. Mapping bit information into binary numbers e.g. maze representation in binary entries. Simple encryption problems.

Data management problems:
File handling utilities and environments. Database construction and management. Operations on data (sorting, searching, shifting, etc.). Stock handling programs. Simple statistical analysis.

Note: This assignment should include a brief description of the functionality and structure of the program. The program should be well-structured, well documented, and evidence of adequate testing should be documented.

5.2. Assignment Marking Scheme

Problem Definition

Presentation and clarity of the problem chosen:
The way the problem is presented and explained to the reader: whether the problem involves a computerization of an existing manual system e.g. a student database or an original application e.g. a game

How well the shortcomings are identified and what are the specifications the new system should have including forecasted limitations and constraints.

[5]

Programming elements

Assignment Design:
The way classes are designed and explained, using standard tools as expected in Data Structures modules using Class Diagrams and Systems Analysis module using ONE Level 0 Context Data Flow Diagram.

[5]

Sub-programs design:
Explanation of sub-programs used using standard algorithms e.g. pseudo-coding or Flow Charts.

[5]

Use of basic JAVA programming elements:
Good use of JAVA programming elements including, use of: primitive data types, variables, pre/post tested loops, conditional & switch statements, methods with and without parameters, arrays, exception handling.

[15]

Algorithms & Logic

Efficient algorithms:
Credit should be given to candidates who design & employ good programming algorithms for sorting, efficient searching techniques and algorithms which make code re-usable and non-redundant

[5]

Flow of application
A good, logical flow of application execution with good data transfer, logical sequence of events, robustness in program structural design to ensure the actual flow of running matches with the intentional design.

[5]

Interface Efficiency
Credit to the interface which allows the easiest and most efficient navigation, shows a good design and is simple in built.

[5]

Object Oriented Principles

Use of programmer's designed Classes and Objects
The level and quality used in designing own classes which create Objects and the way these Classes are integrated to the main application. How well encapsulation is ensured throughout the running of the program.

[5]

Inheritance:
Design and use of Inheritance principles to reduce the redundant code, including normal inheritance and use of abstract classes

[5]

File Handling

Use of files:
Use of appropriate files to store data generated by the application: Object files, text files.

[3]

File Operations:
Operations carried out on files including reading, sorting, appending, and writing to files.

[2]

Application of JAVA API's

Use of JAVA built-in API's and other API's:
The use of JAVA standard API's such as packages (e.g. javax.swing, java.awt, java.util etc..) and their respective classes

[15]

Solution Evaluation and Testing Procedures

Evaluation:
In the form of an overall critical appraisal of the assignment, indicating whether or not the assignment goals have been reached. This should be accompanied with justification for any deviations from the original plan.

[5]

Testing Description:
How well the testing is designed, what strategies are employed and how well the test cases are chosen and presented.

[5]

Evidence of testing:
Evidence and documentation of test results according to test cases with input, output, expected output, and screen shots, showing program execution.

[5]

User's manual:
A concise, but complete, user's manual with clear, annotated screen shots, aimed at non-technical, end users explaining how the application can be installed and used.

[5]

Conclusion & Future Improvements:
This should highlight the benefits of the current system, and any areas in the assignment that could be improved upon in any future iteration.

[5]

5.3 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.

5.4 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 candidate's submitted assignment.