IM SYLLABUS (2023)

COMPUTING **IM 07**
SYLLABUS

Computing	(Available in September)
IM 07	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".

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 organised 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: 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

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.
(Candidates will not be asked to explain floating point representation of numbers)

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, including half-adders and full-adders.

Boolean Algebra

Basic theorems and properties of Boolean algebra
Equivalence, contradictions and tautologies
The following list of laws will be assumed:

1. Commutative law.
   $X + Y = Y + X$; $X.Y = Y.X$
2. Associative law.
   $X + (Y + Z) = (X + Y) + Z$; $X.(Y.Z) = (X.Y).Z$
3. Distributive law.
   $X.(Y + Z) = X.Y + X.Z$

4. De Morgan's law.

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

5. Absorption law.

   $X \cdot (X + Y) = X + (X.Y) = X$

6. Double complement law.

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

7. Laws of tautology.

   (a) $X + X = X$; (b) $X.X = X$;

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

   (e) $X + 1 = 1$; (f) $X.1 = X$;

   (g) $X + 0 = X$; (h) $X.0 = 0$.

Simplification of Boolean expressions using the above laws.

Karnaugh Maps

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

Module 2: Computer Architecture and Assembly Language

Objectives

Students should be able to:

- gain a good understanding of the main components making up the computer system following the Von Neumann architecture;
- 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

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.
Overview of the Organisation 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 (Primary level and Secondary Level Cache) 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 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 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; Subtract the contents of register BX from the contents of register AX, storing the result in AX. Immediate – Example MOV AX,000FH; move the contents specified in the second operand into AX Direct – Example MOV BX,2000H - move the contents of the memory location specified in the second operand into BX *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 3: Operating Systems

Objectives

Students should be able to:

- understand the need for, and attributes of, different types of software;
- understand that system software is the interface between the hardware and user applications;
- describe different types of operating systems and their function;
- describe the different functions of an Operating System.

Content

Software	Explain what is meant by: System software Application software
System software	Understand the need for, and functions of the following system software: Operating Systems (OSs) utility programs libraries translators (compiler, assembler, interpreter).
Types of Operating Systems	Describe the main features of the following types of operating systems: batch processing, single-user (standalone), multi-user (multi-access), multi-tasking and multi-programming, real time control and real time transaction systems. Identify and describe applications that would be suitable to these modes of operation.
Functions of an Operating System *Interrupts*:	Describe the main functions of an operating system Describe the conditions which could generate interrupts. Give an overview of interrupt handling and the use of priorities. Describe the factors involved in allocating priorities. Polling vs interrupts Polled interrupts vs vectored interrupts
Memory management	Describe logical and physical address spaces Describe reasons for memory allocation. Consequences of memory allocation : memory fragmentation and compaction. Memory store protection (to prevent processes from accessing storage allocated to other jobs)
Scheduling	Purpose of scheduling: processor allocation, allocation of devices and the significance of job priorities. Explain how the round robin scheduling algorithm works Explain the three basic states of a process: running, waiting, suspended. Explain deadlock. *(Candidates are only expected to define deadlock)*
File Management	Explain the hierarchical structure of a directory,

	Explain the different ways how to protect files, and describe file attributes.
Providing an interface	Describe the need for and the role of the operating system in providing an interface between the used and the hardware.
Utility software	Explain the need for and use of a range of utility software. Consideration of human computer interaction - Explain the need to design systems that are appropriate to the variety of different users at all levels and in different environments.

Module 4 : Networking and Communication

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 PAN 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 broad knowledge is expected from the students).* The need for standard protocols.
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

Module 5 : High Level Language Programming

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;
- have a good understanding of the fundamental concepts of object oriented programming including objects and classes.

Content

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.
High Level Languages	Introduction to programming paradigms Characteristics of each programming paradigm, including imperative, object-oriented and formal.
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
Expressions and data Types	The objectives of the following include an understanding of Simple java programs that utilise print and println, Java style comments, Primitive data types: char, int, double, and boolean, Operator precedence, Relational operators <, <=, >, >=, ==, != String literals, Expressions composed of primitive data types and, Expressions that mix data types.
Variables and assignments	The objectives of the following include an understanding of The concept of a variable as a named memory location, Variable declarations and initialisations, Assignment and Java's assignment operators: =, +=, -=, *=, /=, and %=, The use of a Scanner object for interactive input, The advantages of using final variables Type compatibility and casting, Pre-increment and post increment (++ x, x++, --x, x--) and Shortcuts (x = x + 10 can be shortcut to x += 10).
Selection and decision	The objectives of the following include an understanding of: Selection as a mechanism for controlling the flow of a program;

	The **if** statement, the **if-else** statement, and **Switch** statement and Nested **if-else** statements.
Repetition/ Iteration	The objectives of the following include an understanding of Repetition and loops: the while, do-while, and for statements; The differences and similarities among the while, do-while and for statements; Nested loops; Infinite loops.
Methods	The objectives of the following include an understanding of: The concept of a method as a "black box"; The methods of Java's Math class* How to construct methods that carry out simple tasks; The differences between void methods (setters) and methods that return a value (getters); Local variables; Method overloading. **Only the following methods are to be used: Math.random(), Math.round(x), Math,max(x,y), Math.min(x,y), Math.pow(x,y), sqrt(x)*
Objects and classes	The objectives of the following include an understanding of: Objects and classes; Programmer defined classes; Components of a class: constructors, instance variables and methods; Overloading of constructors and methods; Access modifiers: public and private; Encapsulation and information hiding; Inheritance; Static variables and static methods; Parameter passing (passing by value or by reference); The keyword *this*
Arrays	The objectives of the following include an understanding of: Array declarations (single and two dimensional) instantiation, initialisation, filling in an array with data, displaying data from an array; How arrays are passed to methods and used in methods; Sorting an array using bubble sort algorithm only; Searching an array using linear search. Array of Objects.
Data structures	Purpose of data structures *algorithms*
Stacks	Basic knowledge of the concept of a stack as a data structure.
Queues	Basic knowledge of the concept of a queue as a data structure.
Linear Lists	Basic knowledge of the concept of lists
Other structures	Arrays – single and two 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 bubble and insertion sort.
Searching Algorithm	Linear Search

Module 6: Systems Design

Objectives

Students should be able to:

- 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

Software systems	The importance of thinking before doing A software development life cycle (simplistic "Waterfall")
Overview of the System	The main stages of a system life cycle: Problem Definition Feasibility Study Requirements Elicitation Analysis Design Implementation Testing Maintenance
Problem definition	What prompts an organisation to develop a new system: current system may no longer be suitable for its purpose; technological developments; current system may be too inflexible or expensive to maintain; required to gain a competitive advantage.
Feasibility study	Preparation of a study containing the scope and objectives of the proposed system.
Requirements Elicitation	Understanding the problem completely through: Interviews; Questionnaires; Inspection of documents; Observation (of existing system and work processes).
System Analysis	Using data flow analysis to describe system function (Data flow diagrams) as the principle analysis tool.
System Design	Modular system structuring Top down and bottom up design approaches The use of pseudo code and/or flowcharts
Implementation	User guides and in-code comments. Change-over techniques (basic idea behind and comparison of): 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 testing data within testing ranges. Comparison between black and white box testing concepts.
Maintenance	Types of maintenance : Adaptive Corrective Perfective
Standard methodology	A top level overview of UML.

Module 7: Databases and Database Management Systems

Objectives

Students 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)	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 normalisation process which is applied on it. *Candidates **are not** expected to normalise databases, but rather understand the concept behind normalisation 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. FURTHER INFORMATION REGARDING THE ASSIGNMENTS

4.1 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.

4.2. 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 computerisation 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. Systems Analysis using ONE Level 0 Context Data Flow Diagram and arising Use-Case and Class Diagrams.

[10]

Sub-program design:
Explanation of sub-programsused which use standard algorithms, e.g. through the use od pseudo-code or flowcharts

[5]

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.

[25]

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.

[5]

Application of JAVA 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

[5]

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]

4.3 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.

The allocated teacher reserves the right to interview any candidate of his/her choice regarding the content of the candidate's submitted assignment.