AM SYLLABUS (2012)

COMPUTING AM 07

SYLLABUS

Computing AM06	(Available in September)
Syllabus	Paper 1(3 hrs) + Paper II(3 hrs) + Paper III(Project)

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. 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 programming language will be used 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 Science 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 various aspects of the Computing discipline possibly with an eye to pursuing any undergraduate degree programme through a fundamental treatment of the technical aspects of larger field of ICT.

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, as indicated by the percentages of the various assessable components of this examination, 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 in all. These questions will be spread over two sections. Section A and Section B. Section A shall exclusively contain one or more simple questions that will directly exercise the acquired programming knowledge of the candidate. Section B shall contain questions from any part of the syllabus and that require short and to the point answers each worth 5 marks. Paper I 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. Paper II 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. Paper III (The Project) carries 20% of the final (total) examination score.

Note Regarding use of calculators:

Calculators may NOT be used n 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

Data Representations

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:

- 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 & floating point format representation
Overflow and underflow
The use of codes to define a character set including ASCII, Unicode

Logic Gates

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
The functional completeness of the NAND and NOR gates

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) $X \cdot 0 = 0$

The simplification of Boolean expressions using Boolean algebra rules and/or Karnaugh maps – up to 4 variable maximum. (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

Memory

RAM Memory chip typical organization

- Data input and output lines
- Address input lines
- Write enable line

The characteristics and application of RAM type memory chips

- Dynamic RAM
- Static RAM

ROM Memory chip typical organization
The characteristics and application of ROM type memory chips

- ROM
- EPROM
- PROM
- EEPROM

Memory Address Map
Memory connection to the CPU (address decoders)

I/O Subsystem

I/O Addressing: Memory mapped vs. Isolated/Separated I/O
Handshaking
Interrupts

- Overview of interrupt handling
- Detecting source of interrupt
- Software polling vs. vectored interrupts

DMA

AGP, PCI and PCI-X buses

Characteristics as follows:

- Throughput described as bits per second;
- Width;
- Multipoint Topology;
- Function.

CPU	CPU model and overview of main components CPU's instruction set and instruction format Detailed description of processor and registers including MAR, MDR, CIR, PC and general purpose registers. • Control Unit • Arithmetic Logic Unit • The fetch, decode and execute cycle in terms of buses and registers. • The stack and its role in subroutine transfer
CPU Registers	The purpose and use of special internal registers in the functioning of the CPU including • Data registers • Segment registers • Index registers • Stack registers • Control registers • Status or flag register • Cache (description of the structured use of cache memory to improve processor performance)
Overview of some CPU design issues	RISC CISC *(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/O Peripherals	Serial, parallel, USB ports and flash RAM. Serial data transmission • 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

Assembly Language Instructions	Instruction sets Instruction format – opcode & operand Mnemonic representation of opcode
Instruction Groups	See Appendix A for limited instruction set
Instruction Formats	*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.*
Registers	The general purpose registers: AX (Accumulator), BX (Base), CX (Count) and DX (Data) as 16 bit registers with reference to their 8 bit high order and low order bytes.
Addressing Modes	Register addressing, e.g. INC AX (increment value of AX by 1) Immediate addressing, e.g. MOV AX, 03H (move value 3 hex into reg. AX) Direct addressing (also known as memory addressing). This refers directly to a memory address and is allows the transfer of data between this memory location and a register, e.g. MOV AX, [0810H] (move into accumulator the contents of memory location 0810H) Indirect addressing, e.g. MOV AX, [BX] (the contents of the BX register is an address and is used to point to the memory location where the data is to be found) Indexed addressing, e.g. MOV CX, [BX + DI] (the value in the base index register BX is combined with the number in the destination index register DI to provide address of the number to be loaded into the CX register)
Assembler	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

Content

Operating Systems

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

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

Job Control Languages (JCL) and the use of JCL. No coding knowledge is required.

The main functions of an OS:

i. Process Control

Process management

States of a process

- Run
- Wait
- Suspend

Scheduling

- Round Robin
- Priority
- Deadlock
- Deadlock avoidance
- Deadlock detection

ii. Memory Management

Memory maps of single and multi programming environments

- Contiguous memory partitioning
- Logical vs. physical address spaces
- Relocate-ability
- Memory fragmentation and compaction
- Pages and page frames
- Size considerations and Faults
- Memory store protection
  (to prevent processes from accessing storage allocated to other jobs

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 rights and allocated privileges
- File attributes
- Hardware failure

iv. Handling of I/O operations

I/O Addressing: Memory mapped vs. Isolated/Separated I/O
Handshaking of Devices

Devices that minimise complexity of I/O operations

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

V. Interrupts Handling

Interrupts

- Overview of interrupt handling
- Detecting source of interrupt
- Software polling vs. vectored interrupts
- DMA

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.

Overview of the Transport Control Protocol/Internet Protocol (TCP/IP)

Point-to-point Connections

Basics of communications

- The concept of sender, medium and receiver
- Simplex (e.g. TV tele-text, radio receiver, etc.)
- Half duplex (walkie talkie, two-way radios, etc.)
- Full duplex (telephone unit, mobile phone, etc.)

Parallel and Serial transmission (synchronous and asynchronous)

Analogue vs. digital signals

Bandwidth, Bit rate and baud rate

Transmission Technologies

Narrow band (dial-up),

Broadband: ISDN, DSL, ADSL

Peer-to-peer, broadcast networks

Modems

Modulation

- amplitude
- frequency
- phase
- pulse coded (PCM)

Demodulation

Multiplexing

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

Transmission media

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

Noise & Interference as two factors which affect data integrity

Computer Networks

Definition, classification and properties of a computer network

- PAN
- LAN
- MAN
- WAN

Network Topologies	The benefits and drawbacks of basic network topologies • Bus • Ring • Star • Mesh (Partially & Fully interconnected)
Media Access Methods	Properties of the CSMA/CD as a means of controlling collisions in Bus networks. Properties of Token passing as a means of managing Ring networks (improvements for FDDI and fast Ethernet)
Switching Techniques	The difference between various switching techniques and their application • Circuit-switching • Message-switching • Packet-switching Datagram = packet of data + auxiliary control data Buffering Use of switches, hubs, repeaters, bridges and routers.
Error checking and Recovery	Transmission errors and methods to overcome them • Parity (single and block) checking • Cyclic Redundancy Checking (CRC) *(only in principle)* • Message acknowledgement (Implicit and Explicit) • Retransmission schemes (Stop and Wait, Go-back-N, Selective Repeat)
IP Addressing	IP as an addressing scheme for every machine on the Internet for successful delivery of information, its use in subnets, DNS and URL translation
Internet Applications	E-learning, distance learning, e-commerce, virtual worlds, conferencing, research, travel, news and communication
Effective Web exploitation	Literature searching Topic tracking and coverage Collaboration methods Plagiarism and professional respect Netiquette in general
Social Implications of Computing and Internet	Social networking 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
New transmission technologies	Attributes of Wireless LAN (WLAN) / WiFi. Overview of modern wireless standards: 802.11 b/g/n. Wireless network technologies: hotspots, wireless routers, wireless repeaters. Overview (non-technical) of the WiMax wireless technology, Internet key/dongle.
Integration of Internet and WWW-related Protocols and Terminology	Hypertext Markup Language (HTML), hypermedia, authoring tools, ADSL, ISDN, Telnet, FTP, IMAP, POP3 *Note: Only familiarity in broad terms (conceptual and comparative only) is required in this section*

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

Content

Formal languages	The differences between natural and formal languages The alphabet of the language Terminal and non-terminal symbols Language productions
Defining the syntax of a programming language:	Defining syntax using BNF and Syntax Diagrams. Overview of EBNF. *Note* that EBNF is not standard and students are only required to have a general knowledge of the EBNF standard.
The syntax of a formal language	Symbol classes (identifiers, delimiters, operators, numbers) Reserved words The use of BNF/EBNF to unambiguously express the syntax of a language The use of parse trees (bottom up / top down) 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 The use of binary trees and stack to obtain and evaluate a post-fix (RPN) statement
The semantics of a formal language	The need for semantics (meaning) other than syntax (form) Context sensitivity
The compilation process	The stages of compilation • Lexical analysis - removal of redundant text, simple error handling - conversion of lexemes to tokens • Syntax and semantic analysis - parsing - symbol table - compile-time error detection and handling • Code optimisation & generation - simple techniques to optimise code - translation into object code, linking
Language translators	The differences between assemblers, compilers and interpreters Other types of compilers: macro pre-processors, 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 main principles of systems analysis and design
- Develop a practical knowledge of the main stages of the systems development life cycle including: identification of problem, feasibility study, information processing requirements, analysis, design, implementation, testing and maintenance.

Content

Overview of the System LifeCycle

The main stages of a system life cycle: Feasibility Study, Analysis, Design, Programming, Testing, Installation and Maintenance.
Compare and Contrast The Waterfall lifecycle with that of Rapid Application Development (RAD).

Identification of the problem

What prompts an organization 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

Understanding the problem completely through:

- Interviews
- Questionnaires
- Inspection of documents
- Observation (of existing system)

Feasibility study

Preparation of a report containing the scope and objectives of the proposed system. The feasibility report should determine whether it is worth proceeding from a number of aspects. The main feasibility aspects to consider are:

- Technical
- Operational
- Timeliness
- Economic
- Legal
- Social

Candidates should be able to describe and reason in the above terms whether or not the project is feasible

System Analysis / Requirements Analysis

System requirements

- Processing required
- Data storage
- Input and output formats

Joint Application Development

- How can JAD help in identifying business requirements?
- In what circumstance is it best to use JAD?

UML diagrams: How can UML help system analyst model various part of the system.

- What is a Use Case Diagram?
- How can a Use Case Diagram help in determining system requirements?
- Class Diagrams and their basic application.

Data flow of system: Fundamental concepts of Data Flow Diagrams (DFDs only up to Level 1)
Entity Attribute Models (only basic use)

Formulation and evaluation of alternative proposals
Development and maintenance needs for each solution
Choice between "off-the-shelf" solutions and purpose-built ones

System Design

Top-down and bottom up approaches to system design

The design specifies:

- Design of user interface
- Menu design
- Specification of data employed (data inputs)
- Organization of data output
- Specification of hardware and software selection
- Conversion plan
- Test strategy, test plan, and test data. (Test cases)

The use of the following algorithm representation forms:

- Hierarchical Input Output Processing (HIPO) chart
- Jackson Structured Programming (JSP) method
- Decision tables
- Flowcharts
- Structured text and pseudo-code

Modular design and modular interface concepts

The use of Prototyping
Comparing the Spiral Model with the Waterfall model

Preparing the documentation

Coding and Testing

Coding of modules
Documentation of any deviations from the original design
Module development
Testing according to Testing strategy
Preparing the User Manual

Types of Testing:

- Bottom up testing
- Top down testing
- Black box and White Box testing
- Alpha and Beta testing

Implementation

Tasks to be faced before the changeover is complete:

- Installing any applicable hardware
- Training system users
- Creation of master files

Change-over techniques (basic idea behind and comparative):

- Direct
- Parallel
- Phased
- Pilot

Maintenance

Best practices in (fundamental points only):

- System analysis
- Modularity
- Documentation generation

Types of maintenance (basic concepts behind and examples):

- Adaptive

- Corrective
- Perfective
- Predictive

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 statements

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 from a conceptual point of view)
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 (supported through examples) Creating a stack (difference between a static and dynamic structure) Push and Pop algorithms to add and delete elements from a stack Traverse stack to display its contents
Linear Lists *Linked Lists* *Circular Lists* *Double Linked Lists* *Queues*	Creating the structures Adding a node to the structure Deleting a node from the structure Traversing the structure
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 and/or Array Lists
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	
Files	Text files, Random files and Object Files • Creating a file

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

Serialization

- The *Serialize-able* interface
- Serializing single instances
- Serializing Vectors or Array Lists

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 the 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.
Relational 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 (stud_id, 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 the following cardinality: • one-to-one, • one-to-many and • many-to-many relationships *The standard "Crow's Foot" notation is to be used to model and describe the above concepts.*
Normalisation	The importance of normalisation to avoid unnecessary redundancy Normalise a simple relational database up to the Third Normal Form
Database Applications	The purpose and use of commercial and top-end database packages, web-based database solutions Develop a simple relational database using fourth generation applications such

	as Microsoft Access™ or Delphi™
Structured Query Language (SQL)	Understand the purpose and use of SQL commands to manipulate data including: SELECT, FROM, WHERE, ORDER BY, HAVING, GROUP BY, JOIN *Candidates will **NOT** be expected to write segments of SQL, only interpretation of SQL instructions will be examined.*

APPENDIX A (TO MODULE 2): ASSEMBLY LANGUAGES

Limited instruction set to be used

Data Transfer instructions	MOV	Moves byte or word to register or memory
	PUSH	Push a word on stack
	POP	Pop a word from stack
Logical Instructions	NOT	Logical not (1's complement)
	AND	Logical and
	OR	Logical or
	XOR	Logical exclusive-or
Arithmetic Instruction	ADD , ADC	Add and Add with carry
	SUB, SBB	Subtract and Subtract with borrow
	INC	Increment
	DEC	Decrement
	CMP	Compare
Transfer Instructions	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
Flag Manipulation	CLC	Clear Carry
	STC	Set Carry
Shift and Rotate	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
FDDI	-	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's basic text book

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

4.2 Other recommended text books

Brooshear, 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 references books

Computer Architecture and Assembly	Abel, P., *IBM PC Assembly Language and Programming*. Prentice Hall.
	Kleitz, W., *Digital and Microprocessor Fundamentals*. Prentice Hall.
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.
Operating Systems	Ritchie, C., *Operating Systems. Incorporating Unix and Windows*. Continuum.
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 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 though 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

School project assessment marks are to be submitted to the MATSEC Support Unit not later than the date stipulated by the MATSEC Board.

Note on Private candidates:
Private candidates are to submit all exercises for assessment to MATSEC Support Unit by the date stipulated by the MATSEC Board.

All candidates may be called for an interview regarding their work.

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 may 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.

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 and how they are to be run. Only work on once-recordable CDs will be accepted. Any handed -in CDs 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	Ariel
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 understand-ability, essential information, subsidiary or detailed technical material should be included in an appendix (recommended not to exceed half the size of the project report).

5.6 Grading Scheme

The overall examination grade will be based on an overall aggregate score, as indicated by the percentages of the various assessable components in Section 2 of this syllabus, and students must obtain a minimum mark in each paper to be established by the Markers Panel. Therefore, both project and written components should be considered as failing. Furthermore, the written component will contain a question, or questions, of a simple nature in compulsory Section A of Paper I which will specifically exercise the basic programming knowledge of candidates. Furthermore, candidates can qualify for Grades A to C, ONLY if they satisfy the examiners in the programming exercise(s) in Section A of Paper I.

Candidates will be allowed to re-submit the project in the next session if they fail to satisfy the examiners in this component of the examination in the first session. All candidates who fail in the project will be informed accordingly.

5.7 Project Marking Scheme (guidelines for project assessors)

The award of marks will be based on the following assessment criteria.

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

Project 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

[8]

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.

[8]

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

[4]

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.

[4]

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

[3]

Polymorphism:
Use of polymorphism in methods and arrays of Objects which suit any form of object

[3]

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

[5]

Solution Evaluation and Testing Procedures

Evaluation:
An overall critical appraisal of the project and whether the aims of the project have been reached or not with justifications for any deviations from the original plan.

[8]

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 the program running.

[8]

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.

[7]

Conclusion & Future Improvements:
The benefits of the current system and any areas in the project that need improvement.

[2]

5.8 Accredited Schools

Schools presenting candidates for this examination must normally offer full-time courses in Computer Science 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.

5.9 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 any, or of all, of the candidate's submitted assignments.