AM SYLLABUS (2020)

COMPUTING AM 07

SYLLABUS

Computing AM07	(Available in September)
Syllabus	Paper 1(3 hrs) + Paper II(3 hrs) + Paper III(Practical tasks)

1. INTRODUCTION

This MATSEC Advanced-level Computing Syllabus has been prepared and compiled in line with previous syllabi, latest computing and 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.

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

Candidates are expected to gain the most benefit from this subject having completed their SEC level Computing.

This syllabus is ideal for those candidates who wish to deepen their understanding of various aspects of Computing possibly with an eye to pursuing studies at undergraduate level.

This document is organised as follows: The next section (2) briefly outlines the contents of the examination itself while Syllabus details are described in detail in Section 3. Finally, detailed information about the coursework is expanded in Section 5.

2. EXAMINATION

The examination shall consist of three parts. All parts shall be set by the MATSEC Board, namely, two written papers each of three hours duration and a set of practical exercises subdivided into three sessions as described further down in this section. The overall grade will be based on an overall aggregate score, as indicated by the percentages of the various assessable components of this examination. Furthermore, candidates can qualify for grades "A" to "C" if they obtain:

I. the overall minimum pass-mark as specified by the MATSEC Board;
II. a pass mark of 40% or more as stipulated in Paper III.

Paper I (100 marks) shall consist of twenty short compulsory questions in all. These questions will require short and to the point answers each worth 5 marks. Paper I carries 40% of the final (total) examination score. This paper is expected to exercise all or most of the fundamental concepts relating to the subject of Computing at Advanced level.

Paper II (100 marks) shall consist of eight long questions of which candidates are expected to answer five. Each question carries 20 marks and candidates are expected to understand precisely what is being asked and demonstrate understanding by answering in some depth. Paper II carries 30% of the final (total) examination score.

Paper III (100 marks) is a set of practical programming tasks. The set of tasks will be held over a period of two years. The first two tasks of duration 2 hours each will take place in the first year and the third task of duration 3 hours will be held in the second year of studies. The actual dates and times of these tasks are fixed by the MATSEC Board. The first two tasks will each carry a value of 30 marks. The third task will have a value of 40 marks. Paper III carries 30% of the final (total) examination score. The time allocated to the various practical tasks must include the effort required to carry out the

tasks and adequate time for the candidates to think and reason about the proposed solution(s). In case of re-sits, candidates will be allowed to re-sit all three tasks within one year.

In the case of school candidates, the Practical Tasks component (a total of three tasks in sequence) will be continuously, i.e. as a sequence of three tasks, assessed by the class tutor. This component will be marked by the class tutor and should be kept under strict confidential cover. In the case of private candidates, the MATSEC Board will be responsible for the assessment of this practical tasks component.

Schools need to have the necessary equipment and software for the administration of the tasks, namely, a computer network set-up with printing facilities accommodating a separate station for each candidate. Each station should have appropriate BlueJ software installed and also incorporate a means of writing the candidate's work to a CD. Each station should not be connected to the Internet. A lab technician should be present during each test. Teachers should not be available during the test. An invigilator will be sent to supervise the practical test session at every school.

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

Note regarding this syllabus: The italic text in this syllabus is mainly intended as guidance to paper-setters and examiners as to the actual scope, depth and detail that is expected from the various topics, when and where applicable.

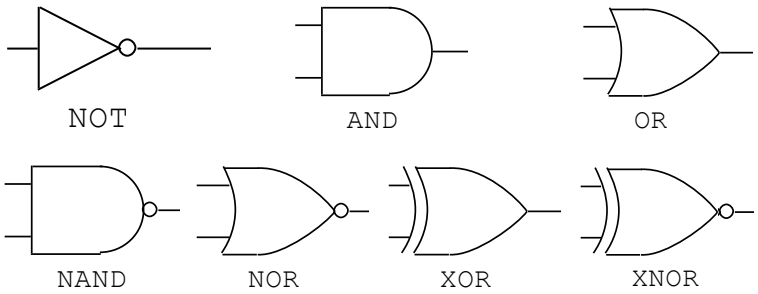
3. SYLLABUS

Module 1: Digital Logic

Objectives

Candidates 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.
- Implement a logic circuit using only NAND or NOR gates.

Subject Content	**Candidates should:**
Binary	Understand that computers use binary to represent data and instructions. Understand how computers represent and manipulate numbers as: 1. unsigned integers; 2. signed integers (sign & magnitude, Two's complement); 3. range of numbers using sign & magnitude, two's complement; 4. binary coded decimal (BCD). *Candidates cannot be asked to perform a representation with numbers exceeding 8 bits.* Be able to perform binary arithmetic as: 1. addition and subtraction, excluding using sign and magnitude; 2. understand the concept of overflow and underflow. Be able to convert between binary and denary whole numbers and vice versa. Understand why hexadecimal notation is used and be able to convert between hexadecimal and binary and vice-versa. Understand the difference between fixed-point and floating-point representation using only 2's complement notation using: 1. signed fractional fixed-point representation; 2. signed floating-point representation; 3. the range of fixed-point and floating-point format representation; 4. normalisation of floating-point numbers. *Questions involving floating-point numbers should NOT exceed 8 bits for the mantissa and 4 bits for the exponent.* *Candidates can be asked to perform normalisation of floating-point positive or negative numbers.*

	Understand the use of code to represent a character set (ASCII, EBCDIC and UNICODE).
Data Storage	Understand and be able to convert between the terms bit, nibble, byte, kilobyte (KB), megabyte (MB), gigabyte (GB) and terabyte (TB).
Logic gates	Understand and define the functions of NOT, AND, OR, NAND, NOR, XOR and XNOR gates, including the binary output produced from all the possible binary inputs (all gates, except the NOT gate, will have 2 or more inputs) Draw truth tables and recognise a logic gate from its truth table. Recognise and use the following standard symbols used to represent logic gates:  Use logic gates to create electronic circuits (combinational logic). Produce truth tables for given logic circuits and convert a truth table to a Boolean function. Produce a logic circuit to solve a given problem or to implement a given written logic statement, such as "IF (switch A is NOT on) OR (switch B is on AND switch C is NOT on) then alarm, X, sounds". Determine the function performed by a combinational circuit through analysis.

Boolean algebra	Understand the basic theorems and properties of Boolean algebra. The following list of laws will be assumed: 1. Commutative Laws a) $A + B = A + B$ b) $A \,.\, B = B \,.\, A$ 2. Associative Laws a) $A + (B + C) = (A + B) + C$ b) $A \,.\, (B \,.\, C) = (A \,.\, B) \,.\, C$ 3. Distributive Laws a) $A \,.\, (B + C) = A \,.\, B + A \,.\, C$ b) $A + B \,.\, C = (A + B) \,.\, (A + C)$ 4. De Morgans Laws *a)* $\overline{(A + B)} = \overline{A} \,.\, \overline{B}$ *b)* $\overline{(A \,.\, B)} = \overline{A} + \overline{B}$ 5. Laws of Absorption a) $A + A \,.\, B = A$ b) $A \,.(A + B) = A$ 6. Double Complement Law a) $\overline{\overline{A}} = A$ 7. Laws of Tautology a) $A + A = A$ b) $A \,.A = A$ c) $A + \overline{A} = 1$ d) $A \,.\, \overline{A} = 0$ e) $A + 1 = 1$ f) $A \,.1 = A$ g) $A + 0 = A$ h) $A \,.0 = 0$ Be able to simplify Boolean algebraic expressions by applying Boolean algebra theorems. *Candidates can be asked to prove the laws using truth tables.*

Karnaugh maps	Simplification of two, three or four variable Karnaugh maps. Simplify Boolean algebraic expressions by applying Karnaugh maps. Understand don't care values and use them to simplify Karnaugh maps *Candidates can be asked to simplify a Karnaugh map using either 1's or 0's.* The applications which should be considered when applying the above theorems (both Boolean algebra and Karnaugh maps) should include all of, and be limited to, the following: 1. Half and full adders; 2. Magnitude comparators; 3. BCD-to-Gray code converters; 4. 7-segment display units (excluding the decimal point). *Candidates can be asked to design a combinational logic circuit of a half and/or full adder.* *Candidates can be asked to design a magnitude comparator given that the numbers being compared do not exceed two bits each.* *Candidates should know how to convert from BCD to Gray code and be able to design a circuit which performs such a function.* *Candidates can be asked to design a circuit to turn on (a) segment(s) of a seven-segment display, given the segments needed to form the range of numbers that can be represented by the seven-segment display.*
Universality of logic gates.	Understand that any logic expression can be implemented using only NAND gates or only NOR gates and no other type of gate. NAND gates alone in the proper combination, can be used to perform each of the basic Boolean operations OR, AND, and NOT. *Candidates can be asked to design a circuit made up of only NAND or NOR gates.*

Module 2: Computer Architecture

Objectives

Candidates 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.

Subject Content	**Candidates should**
Overview of the Organisation of a Computer System	List and describe the main PC components as listed throughout the rest of this module. It should be noted that: 1. The assumed computer architecture follows the Von Neumann model; 2. Candiates should be able to explain the main PC components within the Von Neumann model, including connections with Main Memory and I/O Subsystems through the System Bus.
The System Bus	Be able to explain the use and function of the System Bus as a means of communication between components. The following aspects of Buses should be considered: 1. Address Bus; 2. Data Bus; 3. Control Bus. The following properties/attributes should be understood: 1. Bus size consideration; 2. System clock; 3. Interfacing devices to a common Bus using the method of decoders; 4. Synchronous and Asynchronous data transfer. Be able to give a description of memory read and write cycles. *Bus size considerations must involve the difference in Bus widths for the Data Bus in comparison to the word length, and for the Address Bus in comparison to the addressable space. Size considerations are not to be specific, but general in terms of the comparisons noted above. Questions should be set with a limit of 16 bits in mind and answers can be submitted in index form.*

Memory	Draw the organisation of typical RAM and ROM Memory chips, in terms of: 1. Data input and output lines; 2. Address input lines; 3. Write enable line. Describe and explain the characteristics and applications of RAM type memory chips, in terms of: 1. Dynamic RAM; 2. Static RAM. Describe and explain the characteristics and applications of ROM type memory chips of the type: 1. ROM 2. EPROM 3. PROM 4. EEPROM *ONE application of each case is expected when comparing different types of ROM.* Outline the function of Memory Address Map. Outline the connection between the memory and the CPU through the use of address decoders (the function of address decoders is to be explained and can be drawn in any schematic form that conveys the functional meaning of the decoder). *Applicants may be asked to draw a typical RAM/ROM chip outlining the following components: Chip Select lines; Data Input Lines; Data Output Lines; Control Lines (write/read enable lines); Address Lines.* *The Memory Address Map is a general pictorial representation of how RAM is segmented. The Memory Address Map outlines general processes which are to load into memory, also showing static and dynamic allocation of programs and the difference between these types. Applicants should not be asked to quote specific address ranges for this diagram.*
CPU	Draw the CPU model and provide an overview of its main components. Read and understand the CPU's instruction set and instruction format pertaining to either RISC or CISC architectures as listed in Appendix A of this document. Provide a brief description of registers including MAR, MDR/MBR, CIR, PC and General-Purpose Registers. Candidates may only be asked to diagrammatically represent interactions between the above-listed registers. Explain the function of the Control Unit and the Arithmetic Logic Unit. *Basic knowledge of the function of both units is sufficient.* List the steps involved in the "fetch", "decode" and "execute" cycles in terms of Buses and Registers used. Explain the use of a stack structure and its role in subroutine transfer.

	The instruction set to be followed is that shown in Appendix A of this syllabus. The General-Purpose Registers are limited to the four main ones, i.e. AX, BX, CX and DX. Candidates may be asked to produce a block diagram of processor architecture including the MAR, MDR/MBR, CIR, PC and the Accumulator.
CPU Registers	Explain the purpose and use of special internal registers in the functioning of the CPU including: 1. Data registers; 2. Segment registers; 3. Index registers; 4. Stack registers; 5. Control registers; 6. Status/Flag register; 7. Cache (a description of the basic use of cache memory to improve processor performance, including examples of cache usage). *The discussion of Cache should be limited to Level 1 and Level 2. Comparison of Cache to RAM and Registers should be carried out in terms of access speeds and system performance.*
I/O Peripherals	Be able to describe the following aspects: 1. USB ports and Flash RAM; 2. Serial data transmission; 3. Synchronous transmission; 4. Asynchronous transmission.

Module 3: Assembly Languages

Objectives

Candidates should be able to:
- Understand the general format of assembly language instructions.
- Distinguish between the different types of instruction groups, instruction formats and the various addressing modes.
- Understand basic assembly language programs, given an instruction set.
- List some assembler functions and tools.

Subject Content	**Candidates should:**
Assembly Language Instructions	Understand what an instruction set is. Understand the format of an instruction as being made up of opcode and operand. Understand the mnemonic representation of an opcode. Understand the distinction between an instruction and a pseud-directive.
Instruction Groups	Understand the instruction set given in Appendix A which is based on the instruction set of the 8086 processor.
Instruction Formats	Be able to interpret simple programs written in assembly. *Instruction sets together with relevant descriptions are always to be provided as part of the question. Candidates should be able to interpret fragments of code, but do not need to be able to write code.*
Registers	Understand the purpose of the general-purpose registers inside the processor, including the AX (Accumulator), BX (Base), CX (Count) and DX (Data) registers. *In relation to these 16-bit registers, reference should be made to both their 8-bit high and 8-bit low order bytes, e.g. AH AL.*
Addressing Modes	Understand the following addressing modes: 1. Register addressing, e.g. INC AX (increment value of AX by 1); 2. Immediate addressing, e.g. MOV AX, #03H (move value 3 hex into register AX); 3. Direct addressing (also known as memory addressing). This refers directly to a memory address and allows the transfer of data between this memory location and a register, e.g. MOV AX, 0810H (move int the accumulator the contents of memory location 0810H); 4. 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); 5. 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 Direct Index or Source Index to provide the address of the number to be loaded into the CX register); 6. Understand the meaning of symbolic addressing. *For the sake of clarity, direct addressing should not use square brackets, not to be confused with indirect addressing. Pseudo-directives are not to be counted as instructions.*
Assembler	Understand the assembly process, including the assembling, linking, loading and relocation processes. Understand the purpose of the different types of assemblers, including: cross assemblers, macro assemblers and meta assemblers.

Module 4: Operating Systems

Objectives

Candidates should be able to:
- Describe different operating systems and their function as well as 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.

Subject Content	**Candidates should:**
Types of Operating Systems	Understand the specifics, function and differences of the following types of Operating Systems: 1. Batch; 2. On-line; 3. Real-time; 4. Network. *By "On-line Operating Systems", Operating Systems that would require a working Internet connection to function are meant.* Understand Process Control Operations (definitions only). Understand the application of Job Control Languages (JCL) and the use of JCL. *No coding knowledge is required.*
Process management States of a process	Understand the states in which a process may be, as follows: 1. Run; 2. Wait; 3. Suspend.
Scheduling	Understand how processes can be scheduled, as follows: 1. Round Robin 2. Priority
Deadlock	Understand what deadlock is, how it can occur and how to detect and avoid it.
Memory Management	Understand the following memory usage and issues: 1. Application of Memory Maps of single and multi-programming environments; 2. Contiguous memory partitioning; 3. Logical vs. physical address spaces; 4. Relocate-ability; 5. Memory fragmentation and compaction; 6. Pages and page frames; 7. Size considerations; 8. Faults Memory store protection (to prevent processes from accessing storage allocated to other jobs).
File management	Understand and use the following issues: 1. Management of files as stored physically; 2. Creating and accessing files; 3. Allocation of storage space. Understand the meaning and structure of Blocks in terms of: 1. Contiguous; 2. Linked; 3. Indexed.

	Understand the facilities for editing the contents of files. Understand the protection of files and facilities against their unauthorised access, in terms of: 1. User ID and password; 2. User Home Directory; 3. File access rights and allocated privileges; 4. File attributes; 5. Hardware failure.
Handling of I/O operations	Understand I/O Addressing in terms of Memory mapped vs. Isolated/Separated I/O. Understand the concepts of Handshaking of Devices. Understand devices that minimise complexity of I/O operations as follows: 1. Interrupt vs. polling; 2. Interrupt handler; 3. System stack; 4. Multiple interrupts and interrupt priorities. Understand the Interrupt mask register.
Interrupt Handling	Understand interrupts in terms of: 1. Providing an overview of interrupt handling; 2. Detecting source of interrupt; 3. Software polling vs. vectored interrupts. Understand Direct Memory Access (DMA). *Terms to be examined relating to interrupts are Interrupt Service Request (IRQ); Interrupt Service Routine (ISR); Interrupt Register; Interrupt Enable/Disable Register; Software Polling; Vectored Interrupts – including the Vector Table.*

Module 5: Networking and Communications

Objectives

Candidates should be able to:
- Understand the basics of transmission methods in communication.
- Reason about the concepts behind the data transmission technology.
- Have a clear understanding of protocols, IP addressing properties and differentiate between IPv4 and IPv6.
- Gain a general understanding of the OSI seven-layer protocol.
- Differentiate between LAN and WAN, including an in-depth understanding of the different network topologies in use.
- Understand the implications of a network security breach.
- Gain adequate basic knowledge in the field of data and network hardware security technologies.

Subject Content	**Candidates should:**
Introduction to Network	Be able to: 1. Define the term "computer network"; 2. Understand point to point data communications; 3. Distinguish between simplex, half duplex and full duplex data communications. *Candidates must be able to define each type of data communication and provide examples in each case.* Be able to differentiate between serial and parallel data communication and know where each technology may be used. *Candidates must compare the basic difference(s) between serial and parallel data communication and show where each data communication is more suitable.*
Introduction to Network Data transmission technology	Be able to differentiate between analogue data communication and digital communication (Diagrams being important to help explain differences), and understand bit rates and data sampling. *Candidates should know what bit rate and data sampling is and how these two factors affect the quality of data communication when modulating data.* Understand the use of modems in data communication and what analogue wave modulation and demodulation is as well as the importance of modulation when transmitting data. *Candidates are also expected to know the difference in wave properties of modulation in terms of amplitude, frequency, phase and pulse.* Be able to define modulation and know the reason behind wave modulation and how it can increase data speed transmission as well as the effects of baud rate and bandwidth in the speed of data communication. *Candidates should know the difference between the measures of bit rate and*

	baud rate. Understand the importance of data integrity during data communication and the importance of multiplexing and de-multiplexing. Be able to differentiate between time division and frequency division multiplexing (i.e. TDM and FDM). *Simple questions may be asked to provide diagrams of data transmission using multiplexing. Apart from their definitions, candidates must also be able to explain how to merge multiple data communication on a single line using diagrams, as well as how to reverse the process. Candidates should know the advantages and disadvantages of both TDM and FDM together and the application of each.*
Network Media	Be able to compare and contrast between the following network media types: 1. Coaxial; 2. Twisted pair (including shielded and unshielded); 3. Fibre optic; 4. Wireless (Satellite, Bluetooth and WIFI). *Candidates should know the characteristics of the above (e.g. max length, speed, immune or not to Electro-Magnetic Interference) and the ideal situation in which they should be used.* To define noise and Electro-Magnetic Interference (EMI). *The candidate should know which types of EMI exist and how one can reduce their presence in data transmission.*
Error Detection and error Correction	Be able to name factors which lead to errors in data transmission such as EMI and Noise. *The candidate should define EMI and noise and how these can be mitigated.* Understand the use of Parity and Checksum checks as error detection and correction measures designed to protect the integrity of data, during data transfer. *A polynomial approach to the way CRC works is not required. There is no need to know how to perform a binary division. The candidate should know when and where best to apply each method of error detection.*

IP addressing and Websites	Understand the Internet Protocol address (IP address) in terms of: 1. What an IP address is; 2. Where IP addressing is used; 3. The need for Public IPs and private IPs; 4. IP security issues; 5. Length and format of both types of IP addresses; 6. The disadvantages of the IPv4 format; 7. The IPv6 packet format (the header part only); 8. The IPv6 addressable space. Understand the difference between IPv4 and IPv6 protocols. *Candidates should know at least three IPv4/IPv6 format differences (e.g. addressability, routing, security, etc.) and should understand the concept behind creating IPv6 IP addresses.* Understand what a URL and what DNS is. *Candidates should understand the uses of the DNS and why DNS is useful (i.e. the advantages of DNS as opposed to direct numeric addressing). Candidates should also understand the process of how a URL is converted to an actual web address.*
IP addressing and Websites Protocols	Know what a protocol is and its importance in connecting to different types of network. Tackled Internet protocols should include the following: 1. HTTP; 2. HTTPS; 3. FTP; 4. POP3; 5. SMTP; 6. IMAP. *Candidates are to define what a network protocol is and the importance of network protocols and the area of application of the protocols. Comparing and contrasting these protocols is important. Candidates should also know the advantages and disadvantages of each protocol.* Understand the OSI seven-layer protocol and define its purpose. *Candidates should know the basic function of each layer and inversely, name the layer required to perform a particular network communication function.*

Network Topologies	Understand the types of network: 1. Personal area network (PAN); 2. Local area network (LAN); 3. Metropolitan area network (MAN); 4. Wide area network (WAN). *Candidates should be able to compare each type of network and have a basic idea of each. Candidates should be able to differentiate the usage between PAN, LAN, MAN and WAN in terms of size and technology. There is no need to know the details of technologies used in such networks.* Be able to compare and contrast the following network topologies: 1. Bus; 2. Star; 3. Ring; 4. Mesh; 5. Point-to-Point. *Candidates should know what each topology is and how it is structured and be able to produce diagrams explaining, and list advantages and disadvantages of, each topology.*
Data transmission	Be able to define what CSMA\CD is as well as what a Token Ring is. Candidates should know why and where these two topologies are used. Understand the differences between CSMA\CD and CSMA\CA. *Candidates must know how CSMA\CD works in terms of 1-persistent and non-persistent protocols.* Be able to explain what Datagrams are. Explain simple Datagrams in switching protocols. Understand different types of network switching protocols (i.e. Packet-switching, message-switching, and circuit-switching). *Candidates should know the difference between each network switching protocol and where each switching protocol is best used. Candidates should be able to provide one example of each protocol.*
Connectivity devices	Know the purpose of connectivity devices in networks. Be able to explain when and where the below-mentioned technologies are used in a network. The main characteristics of these connection devices and their application: 1. Routers; 2. Gateways; 3. Hubs; 4. Switches; 5. Bridge.

Security of Information and Infrastructure	Understand what Network Security is and the need to distinguish between both logical (data, information and software) and physical (hardware) security. *The candidate should know at least two logical and two physical network security measures.* Understand the importance of security during data communication. Be able to define encryption and distinguish between Public and Private Keys. Understand the use of digital signatures and digital certificates. *Candidates should only know definitions of keys and certificates and how these can help security in data communication over a network. Candidates must know the process of Signing, Validating and Encrypting a message when using secure sites.* Data protection (confidentiality, integrity and availability) and software resilience to hacking. *Apart from protection of data, candidates must also be aware that software needs to be well-tested so that solutions are not hacked, or provide back doors, etc.* Understand access rights (User Accounts, Groups and Permissions). Understand Firewalls, their use and how they work. *Internal algorithmic and mathematical functioning details of firewalls are not required.* Understand physical threats (natural, intentional and unintentional) and countermeasures adopted (Biometrics, Fire Alarms, etc.) to protect network system hardware. *Candidates should know the most common physical threats to computing resources (both hardware and software) as well as measures adopted in response to these threats.*

Module 6: Language Translators

Objectives

Candidates 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 a formal language and describe the stages of its compilation.
- Differentiate between various types of language translators.

Subject Content	**Candidates should:**
Formal languages and syntax definition	Understand: 1. The differences between natural and formal languages; 2. The syntax and semantics of a language; 3. The meaning of terminal and non-terminal symbols; Know what goes into the production of specific programming languages. Define syntax using BNF and Syntax Diagrams using circles to denote terminal symbols and rectangles to denote non-terminal symbols. Understand the meaning and use of meta-symbols and variables as follows: **::=** defined as; **< >** non-terminal symbol; **|** selection.
The syntax of a formal language	Be able to: 1. Use BNF to unambiguously express the syntax of a language; 2. Be able to use 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; 3. Be able to use Reverse Polish Notation (RPN) to define arithmetical statements; 4. Use of binary trees and stacks to obtain and evaluate a post-fix (RPN) statement. *Assessing of Language Translators sub-topics should be done in consideration of the overall process of program translation. Questions therefore should not focus on sub-topics in isolation from the larger picture. Usage of a stack structure is a must for notation conversion from post-fix to in-fix, while tree structures are a must for notation conversion from in-fix to post-fix.* *Example: A candidate is asked to perform a tree traversal (post-fix to RPN) after asking the reason behind post-order traversal.*

The semantics of a formal language	Understand the need for semantics (meaning) other than syntax (form).
The compilation process	Understand the stages of compilation, as follows. Lexical analysis: 1. removal of redundant text; 2. simple error handling; 3. conversion of lexemes to tokens. Syntax and semantic analysis: 1. parsing; 2. symbol table; 3. compile-time error detection and handling. Code optimisation and generation: 1. simple techniques to optimise code; 2. translation into object code and linking. *Questions on stages of compilation should not only include theory but should also include examples. Therefore, it is not enough to have the candidate explain the stage of code optimisation and generation but should be supplemented with an example of code where the candidates actually show how simple code optimisation is done.*
Language translators	Understand the differences between assemblers, compilers and interpreters. Know other types of compilers: macro pre-processors, cross-compilers, p-code compilers. Understand virtual machine concepts and just-in-time compilation.

Module 7: Systems Analysis and Design

Objectives

Candidates 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 (SDLC), following the Waterfall methodology, including: identification of problem, feasibility study, information processing requirements, analysis, design, implementation, testing and maintenance.
- Analyse a given scenario and apply the SDLC stages resulting in the definition of outcomes from each stage in the form of a milestone report.

Subject Content	**Candidates should:**
Overview of the System	Understand the main stages of a system life cycle: 1. Problem Definition; 2. Feasibility Study; 3. Requirements Elicitation; 4. Analysis; 5. Design; 6. Implementation; 7. Testing; 8. Maintenance; 9. Retirement.
Lifecycle	Understand the Generic Waterfall model.
Identification of the problem	Understand what prompts an organisation to develop a new system: 1. Current system may no longer be suitable for its purpose; 2. Technological developments; 3. Current system may be too inflexible or expensive to maintain; 4. New system required to gain a competitive advantage.
Feasibility study	Be able to prepare a study containing the scope and objectives of the proposed system. This study would determine whether it is worth proceeding from a number of aspects, as follows: 1. Technical; 2. Operational; 3. Timeliness; 4. Economic; 5. Legal; 6. Social. *Candidates should be able to describe and reason in the above terms whether or not the project is feasible to develop.*
Requirements Elicitation	Understand the problem completely both in breadth and in depth in terms of: 1. Interviews; 2. Questionnaires; 3. Inspection of documents; 4. Observation (of existing system and work processes); 5. Consideration of the use of "off-the-shelf" solutions and purpose-

	built development. Formulation and evaluation of alternative solutions following the establishment of system requirements. *Candidates should be able to list ONE advantage and ONE disadvantage whilst evaluating each of the above-listed methods.*
System Analysis	Understand and carry out system modelling in terms of: 1. Process modelling; 2. Data Flow based modelling (Data Flow Diagrams - DFDs); 3. Use-Case-based modelling (Use-Case Diagrams - UCDs); 4. Class Diagrams; 5. Data processing; 6. Entity Relationship Diagrams (ERDs). *With the exception of Unified Modelling Language (UML) diagrams, questions relating to the construction of models using the above-listed diagrammatic notations should be restricted to completion of existing models and not include construction from scratch. Questions relating to all the above should not exceed diagram configurations of five (5) nodes. Data Flow Diagrams should not exceed the detail of a Level 1 diagram.* Understand and use UML diagrams in terms of: 1. How can UML help the system analyst model various parts of a software solution? 2. What is a Use-Case Diagram (UCD)? 3. How can a UCD help in determining system requirements? 4. What are Class Diagrams and what is their basic use? Understand the advantages of using UML diagrams. *Candidates should be able to create a UML and/or DFD diagram for a given scenario (max five (5) classes and/or equivalent processes) and understand and explain a given UML diagram and/or DFD.* *The use of the De Marco notation is expected. Please refer to Appendix C for the specific De Marco notation.* Entity Relationship Models (only basic use) *The use of the "Crow's Foot" notation is expected. Please refer to Appendix D for the specific "Crow's Foot" notation.*
System Design	Understand top-down and bottom up approaches to system design. Understand specific design aspects in terms of: 1. Design of user interface; 2. Menu design; 3. Specification of data employed (data inputs); 4. Organisation of data output; 5. Specification of hardware and software selection; 6. Conversion plan; 7. Testing strategy, test plan, and test data (i.e. test cases design). Understand the use of the following algorithm representation forms:

	1. Hierarchical Input Output Processing (HIPO) chart; 2. Jackson Structured Programming (JSP) method; 3. Decision tables; 4. Flowcharts; 5. Structured text and pseudo-code. Be aware and understand modular design and modular interface concepts. *Candidates should be able to list THREE advantages of using a modular design.* Understand the uses of prototyping. Know what goes into preparing software solution documentation.
Coding and Testing	Understand: 1. The coding of modules and the documentation of any deviations from the original design. 2. Module development and testing according to set Testing strategy. 3. The preparation of a User Manual. Understand the types of testing strategies as follows: 1. Bottom up; 2. Top down.; 3. Black box and White Box; 4. Alpha and Beta testing. *Candidates should be able to explain each of the testing techniques and identify and explain which would be appropriate in a given scenario*.
Implementation	Understand the tasks that need to be faced before the changeover is complete, i.e. installing any applicable hardware, training system users and the creation of master files. Know changeover techniques (basic idea behind and comparison of): 1. Direct; 2. Parallel; 3. Phased Pilot.
Maintenance	Be aware of the basic fundamental issues of best practices adopted in system analysis, modularity, and documentation generation. Understand the types of maintenance (basic concepts behind and examples) as follows: 1. Adaptive; 2. Corrective; 3. Perfective; 4. Predictive. *In order to exam the candidates' knowledge on this module, it is advised that a business case/scenario is given listing the requirements for a new/improved update to a system. Candidates are then to create a report-like review listing the stages following the Waterfall methodology (or those parts of the methodology depending on the total marks allocated for this question as specified within the question itself) whilst keeping in mind the*

	requirements listed. Any assumption(s) made by candidates need(s) to be stated. *Short theoretical questions may also be listed, however it is advised that such questions be related to the given scenario.*

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

Objectives

Candidates should be able to:
- Identify and describe different programming paradigms including imperative, declarative, functional and object-oriented.
- 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 (to be implemented using pseudo-code or structured text).
- Know how to make use of files as a permanent type of storage mechanism.

Subject Content	**Candidates should:**
Data Structures	Understand the purpose and applications of the following data structures: 1. Arrays; 2. Lists (in the form of Stacks, Queues, Linked-lists and Array or Linear lists); 3. Binary Trees. *More details regarding these structures are found further down in this module.*
Arrays	Understand what an array is and its purpose. Be able to write algorithms: 1. to create an array; 2. to fill an array with primitive and string type only; 3. to display data from an array.
Pointers	Understand what a pointer is, and what it is used for in conjunction with different data structures. *Candidates are only expected to describe what the use of a pointer is.*
Stack	Understand what a Stack is in terms of a LIFO structure. Be able to perform operations on Stack structures, limited to "Pushing" and "Poping" items on a Stack. Be able to write algorithms using pseudo-code for the "Push" and "Pop" operations. Explain the use of a stack in computing.
Queue	Understand what a Queue is in terms of a FIFO structure. Distinguish between a Linear Queue and a Circular Queue.

	Be able to write algorithms in pseudo-code to add and remove an item from both types of Queues. Explain the use of a Queue in computing.
Lists	Understand the difference between a linear (or array lists), linked list, circular linked list and double-linked list. Be able to describe how to add and delete a node from the above-mentioned lists. *Candidates may be asked to implement the operations on Linear lists using either pseud-code or Java code, but not for linked lists, circular linked lists and double-linked lists.*
Binary Trees	Be able to construct a binary tree. Be able to add and delete a node from the binary tree. Be able to traverse a binary tree using the following three traversal methods: 1. Pre-order traversal; 2. In-order traversal; 3. Post-order traversal. Be able to use a binary tree and an associated traversal method to sort data *Candidates should NOT be asked to use either pseudo-code or Java code to implement the operations on a binary tree.*
Static vs Dynamic data structures.	Be able to explain the difference between static and dynamic implementation of data structures, highlighting the advantages and disadvantages of each. Explain how an array may be used to implement a Queue and a Stack structure
Hash table	Understand what a hash table is. Be able to create and update a hash table using a hashing algorithm. *Candidates can only be asked to use the modulus method to create a hash address.* Understand the concept of collisions and how they can be resolved. Be able to list the characteristics of a good hashing function.
Searching algorithms	Explain the difference between a binary search and a sequential search, highlighting the advantages and disadvantages of each Be able to write an algorithm to search for an item in a sequential list. Be able to show understanding of the conditions necessary for the use of a binary search. Be able to write a binary search algorithm to search for an item in an ordered list. Understand how the performance of a binary search varies according to the number of data items.

Sorting algorithms	Be able to write algorithms to implement the bubble, insertion and selection sort. Be able to dry-run an algorithm for the Quick and Merge sort.
High Level Languages	Understand programming paradigms in terms of: 1. Characteristics of each programming paradigm including: imperative, declarative, functional, object-oriented and event-driven programming; 2. Domain relevance of the above-mentioned programming paradigms. Be able to compare between Object-oriented and Imperative Programming in terms of: 1. The need for a programming paradigm which models the real world in terms of software reusability; 2. The limitations of imperative programming: variable assignment rather than object manipulation; 3. The object-oriented solution: the use of classes and objects in problem-solving. Understand Object-Oriented Programming Characteristics in terms of: 1. Encapsulation (through classes and objects including attributes and operations); 2. Message passing (i.e. operation invocation); 3. Inheritance; 4. Information hiding; 5. Polymorphism.
Expressions and data types	Understand the following: 1. Simple java programs that utilise the keywords **print** and **println**; 2. Java style comments; 3. Primitive data types: char, int, double, and Boolean; 4. Operator precedence; 5. Relational operators <, <=, >, >=, ==,!= 6. String literals; 7. Expressions composed of primitive data types; 8. Expressions that mix data types.
Variables and assignments	Understand the following: 1. The concept of a variable as a named memory location; 2. Variable declarations and initialisations; 3. Assignment and Java's assignment operators: =, +=, -=, *=, /=, %= 4. The use of a Scanner object for interactive input; 5. The advantages of using final variables; 6. Type compatibility and casting; 7. Pre-increment and post increment (++ x, x++, --x, x--); 8. Shortcuts (x = x + 10 can be shortcut to x += 10).
Selection and decision	Understand the following: 1. Selection as a mechanism for controlling the flow of a program; 2. The **if** statement, the **if-else** statement; 3. **Switch** statement and Nested **if-else** statements.
Repetition/	Understand the following:

Iteration	1. Repetition and loops: the **while**, **do-while**, and **for** statements; 2. The differences and similarities among the **while**, **do-while** and **for** statements; 3. Nested loops; 4. Infinite loops.
Methods	Understand the following: 1. The concept of a method as a "black box"; 2. The methods of Java's Math class*; 3. How to construct methods that carry out simple tasks; 4. The differences between void methods (setters) and methods that return a value (getters); 5. Local variables; 6. 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	Understand the following: 1. Objects and classes; 2. Some typical methods in the String class: *charAt, compareTo, concat, equals, indexOf, length;* 3. Programmer defined classes; 4. Components of a class: constructors, instance variables and methods; 5. Overloading of constructors; 6. Access modifiers: public, private and protected; 7. Encapsulation and information hiding; 8. Static variables and static methods; 9. Parameter passing (passing by value or by reference); 10. The keyword **this**
Arrays	Understand the following: 1. The static nature of an array; 2. Array declarations, instantiation, initialisation, filling in an array with data, displaying data from an array; 3. Reference variables; 4. How arrays are passed to methods and used in methods; 5. Sorting an array ONLY using the bubble sort algorithm; 6. Searching an array using a linear and a binary search; 7. Two dimensional arrays *(populating with primitive data types and displaying);* 8. Arrays of objects.
ArrayList	ArrayList *(the dynamic nature of Arraylists, their usage).* ArrayList methods: *add, remove, clear, get, set, size, indexOf, isEmpty, lastIndexOf, removeAll.*
Recursion	Understand the following: 1. Recursion as a method of program control; 2. Problem solving via recursive thinking; 3. Tracing and debugging recursive method. *Candidates will not be expected to write recursive code in paper 3, however theoretical understanding and/or a dry run may be requested in*

	the other papers.
Inheritance	Understand the following: 1. Inheritance and its benefits; 2. The *is-a* relationship between a derived class and a base class; 3. Abstract classes and methods and inheriting from these; 4. Method overriding.
Polymorphism	Understand the two types of polymorphism (method overloading and upcasting). *Candidates are expected to know how to implement a simple polymorphism exercise using an array of a superclass.*
Exceptions	Understand the usage of the *try…catch* construct in data input validation. *Candidates should not be asked to write throwable exceptions.*
Files and File Access	Understanding Text files and Object Files in terms of: 1. Creating a file; 2. Writing to a file; 3. Reading from a file.

Module 9: Databases

Objectives

Candidates should be able to:

- Understand the basic structure, function and importance of database management systems (DBMSs)
- 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.

Subject Content	**Candidates should:**
Database Management Systems	Understand the structure and function of database management systems (DBMSs) as follows: 1. Data dictionary; 2. File manager; 3. Data manipulation language (DML); 4. Data description language (DDL); 5. Query language; 6. Security; Understand the responsibilities of a database administrator. *Candidates should be able to describe the three levels of the database schema.*
Database Models	Be able to compare flat files, hierarchical, network and relational database models, and object-oriented database models.
Relational Databases vs. Traditional File Systems	Understand: 1. The advantages of databases over traditional file systems including: improved data consistency and portability, control over data redundancy, and greater security. 2. The disadvantages of databases over traditional file systems including: greater complexity and cost, vulnerability to system failure and unauthorised access, and larger size.
Relational Databases	Understand the nature and logical structure of a relational database as a set of tables linked together using common fields. Understand the purpose of primary, secondary and foreign keys, attributes (field), and tuples (record). Be able to 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. CANDIDATE (stud_id, name, surname, DoB, address)

Entity-Relationship Modelling	Understand the use of Entity-Relationship (E-R) Models to give a graphical description of the relationship between entities, including cardinality. *The standard "Crow's Foot" notation is to be used to model and describe the above concepts.* Understand the importance of normalisation to avoid unnecessary redundancy. Be able to normalise a simple relational database up to the Third Normal Form. The purpose and use of commercial and top-end database packages, and web-based database solutions. *Candidates should only be aware of fourth generation applications used to develop a database, such as Microsoft AccessTM.*
Normalisation Database Applications	Be able to use E-R Models to give a graphical description of the relationship between entities, including cardinality. Be able to develop a simple relational database using fourth generation applications such as Microsoft AccessTM or DelphiTM.
Structured Query Language (SQL)	Understand the purpose and use of ONLY the following SQL commands to manipulate data: *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

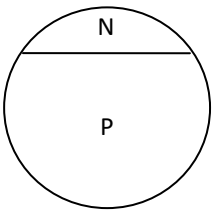
Pseudo-directives	HALT, END

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

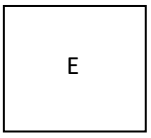
APPENDIX C: De Marco DFD Notation

Data Process:



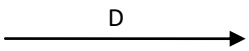
"N" is the process
"P" is the process

External Entity:



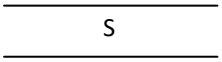
"E" is the entity's

Data Flow:



"D" is the flow label

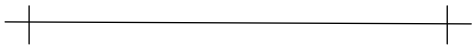
Data Store:



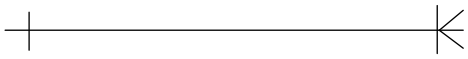
"S" is the Store's

APPENDIX D: Crow's Foot E-R Diagram Notation

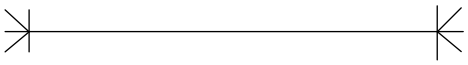
1 – 1



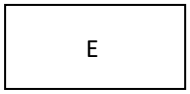
1 – M



M – N



Data Entity



"E" is entity's name

4. FURTHER INFORMATION REGARDING THE PRACTICAL TASKS

IMPORTANT NOTE: *Only the "BlueJ" programming environment will be used throughout in the practical tasks.*

5.1 Rationale

- Problem solving;
- Programming skills;
- Object-oriented concepts;
- Solution extraction and selection;
- Assimilation of algorithmic constructs;
- Expression of problems/tasks using stipulated syntax;
- Applicative skills.

5.2 Practical Tasks component content

All listed below topics are subject to syllabus scope. (Please refer to module 8)

Paper IIIa (year 1, session 1)

1. Identify, understand and use the:

a. basic programming constructs
   i. variables (Class instantiation is not included)
   ii. data types
   iii. sequence statements
   iv. conditional statements
   v. repetition statements

Questions in this task are expected to be three disjoint from each other questions of 10 marks each.

2. Paper IIIb (year 1, session 2)

a. Methods
b. Method overloading
c. Access modifiers
d. Arrays of Primitive data type
e. Linear search
f. Bubble sort

Questions in this task are expected be three in all. The first two questions will be linked to each other in form of question two using the results from question 1, and one third question will be disjoint from the first two. Each question will carry 10 marks.

3. Paper IIIc (year 2, session 3)

a. Arraylists
b. Arrays of objects
c. Inheritance / Abstraction
d. Method overriding
e. Polymorphism
f. Handling exceptions (input mismatch exception and array index out of bounds)

This task is expected to be made up of one question composed of various parts. All parts will be linked and each part will be marked separately. The task as a whole will carry 40 marks.

5.3 Note on private candidates regarding registration for Practical Tasks

Private candidates are to contact MATSEC, who will subsequently allocate each candidate to a specific centre. Registrations will be accepted up till the end of November of their first year of studies. The same assessment conditions and times apply to both school and private candidates.