SEC SYLLABUS (2016)

COMPUTING SEC 09

SYLLABUS

Computing SEC 09

(Not available in September)

Syllabus

Paper I (2 hrs) + Paper II (2 hrs) + Coursework

This syllabus assumes that schools and other educational establishments preparing candidates for this examination will provide suitable facilities for practical computer experience in:

- Using an operating system with a graphical user interface;
- Using a variety of standard application packages – as a minimum a word processor, a database, a spreadsheet and a graphics package;
- Using a variety of standard Internet client packages – as a minimum a Web Browser and an e-mail package;
- Writing, debugging and testing programs in a high level language (Java).

The syllabus is designed to be covered over a period of three scholastic years typically with four lessons of 45 minutes per week. The total hours covered is 240 hours. The practical part could take up some 140 hours of the whole course.

Visit the MATSEC website for resources (http://www.um.edu.mt/matsec/respurces/)

Aims

This syllabus aims to:

1. stimulate and foster an interest in the use of computers;
2. develop practical skills in the use of computers;
3. develop skills in creatively applying information processing technology to problem solving;
4. provide a broad view of the range and variety of computer systems and applications;
5. develop the ability to communicate and interpret information and concepts relevant to computing;
6. introduce students to the fundamental concepts of computer science;
7. serve as a basis for further studies in Intermediate and Advanced level in Computing or Information Technology.

Structure of the syllabus

The syllabus is organised into five main parts, as follows:

1. Computer Applications
2. Computer Architecture and Data Representation
3. Computer Systems
4. Algorithmic Problem Solving and Programming
5. Information and Communications Technology in Society

Sequence of the syllabus does not necessarily dictate the order in which topics are to be taught.

Core and extension topics: The syllabus is further divided into a core component and a set of extension topics. The extension topics are marked with an * and are printed in *italicised* text in the syllabus. Questions about extension topics will be limited to paper 2A.

Supplementary notes: The supplementary notes are included in the right column next to the syllabus and they contain guidelines for teachers and elaboration of the syllabus content. These notes are to be considered an integral part of the syllabus.

Course work (Appendix 2): Appendix 2 gives details about the coursework assessment criteria which will guide Teachers' assessment and the Markers' Panel during the moderation.

ASSESSMENT OBJECTIVES

The examination tests the candidates' ability to:

- recall and understand information and concepts;
- analyse simple problem situations and evaluate feasible computer solutions;
- apply their knowledge of computing to solve simple problems in both familiar and unfamiliar situations;
- keep abreast with current developments, trends and issues related to everyday use of computers.

SCHEME OF ASSESSMENT

The examination will consist of two written papers of two hours duration each, and an assessment of practical work. The maximum overall mark that may be obtained is 200. The questions will be set in English and must be answered in English. Flowchart templates may be used in both written papers. Calculators may **not** be used in either paper.

Questions in the written examination papers will be set to test the cognitive skills – Knowledge, Comprehension, Application, Analysis, Synthesis and Evaluation. The following table shows the distribution of marks of each examination paper for the skill being tested.

Skill Area	**Paper 1 (max 85 marks)**	**Paper 2A (max 85 marks)**	**Paper 2B (max 85 marks)**	**Coursework (max 30 marks)**
Knowledge	30%	20%	35%	-
Comprehension	30%	20%	30%	-
Application	20%	30%	20%	-
Analysis	10%	20%	10%	-
Synthesis	5%	5%	5%	-
Evaluation	5%	5%	-	-
% of Global Mark	40%	45%	45%	15%

Paper 1 (2 hours, maximum mark 85)

This paper will consist of between 10 and 15 short, compulsory questions covering the **core** syllabus. All questions are to be answered in the space provided on the examination paper itself. **All** candidates registered for the examination must take this paper.

Paper 2 (2 hours, maximum mark 85)

This paper consists of five longer compulsory questions. All questions are also to be answered in the space provided on the examination paper itself. There will be two versions of this paper - option A and option B. Candidates are required to indicate on the registration form which option they wish to sit for.

Paper 2A

Questions in this paper will be more difficult than those in Paper 1, and will cover BOTH the core topics and the extensions. Candidates opting for this paper may qualify for grades 1, 2, 3, 4 or 5. The results of candidates who do not qualify for at least a grade 5 shall remain Unclassified (U).

Paper 2B

Questions in this paper will be easier than those in Paper 1, and will cover only the core topics. Candidates opting for this paper may qualify for grades 4, 5, 6 or 7. The results of candidates who do not qualify for at least a grade 7 shall remain Unclassified (U).

Coursework (maximum mark 30)

The coursework exercise listed in **Appendix 2 - Coursework** should be completed by the candidates during their normal course of study under the supervision of their tutor or tutors. There shall be one type of exercise for both Paper 2A and Paper 2B candidates. Candidates should note that:

- The coursework MUST be word processed. Marks for overall layout and presentation are allocated in the coursework marking scheme.

- The mark for the coursework shall be out of 30. 26 marks are allocated for analysis, coding, testing and evaluation of the programming exercise, while the remaining 4 marks for the layout and presentation.
- Excessive length will NOT contribute towards the final mark.
- Tutors should use the Coursework Marking Scheme included in Appendix 2 when marking the exercise. For each candidate, schools or tutors should keep:
    - a copy of the marking scheme, showing the mark awarded by the tutor for each section of the exercise;
    - the candidate's exercise.

Electronic media are NOT to be included with the coursework exercise. The marker's panel set up by MATSEC will be responsible for moderating the tutor-assigned marks and candidates may be called for an interview relating to their coursework.

Private Candidates
Candidates who have never studied the subject at school but have covered the coursework privately will be expected to present the coursework to the MATSEC Board when instructed to do so by the board. Candidates may be called for an interview about their work.

THE PROGRAMMING LANGUAGE

In line with the objective clearly set in the previous version of this syllabus, the programming language that is to be used throughout this syllabus is to be Java. This language will displace the use of the Pascal programming language in every aspect of this syllabus whenever the use of a programming language is required, including the implementation of the coursework.

Transition clause applicable for the year 2013 only regarding the coursework

This transition clause applies to candidates who sat for the SEC Level Computer Studies up to 2012 and are re-sitting the subject:

- Candidates who sat for the SEC Level Examination prior to or in the year 2012 and are re-sitting the subject may carry forward the coursework mark from a previous session, even though the coursework setup and the programming language were different. Notwithstanding this, candidates must still answer questions in the written papers in Java. These candidates are required to fill in a form indicating their request.
- Candidates who have set for the SEC Level Examination prior to or in the year 2012 and are re-sitting the subject may opt to re-submit the coursework provided that the language is Java and the coursework follows the new assessment criteria in Appendix 2.

GRADE DESCRIPTIONS

The following grade descriptors give a general indication of the level of attainment reached by a candidate for SEC Computing. The descriptors are related to syllabus content and not designed to define that content.

GRADE 7

Candidates:

- show a basic knowledge and understanding of the wide-range use of computers today in information processing.
- should show basic ability in using common office application software along with communications software.
- show basic knowledge of computer hardware and peripherals.
- show good knowledge of how the various components function and communicate together as a complete system.
- should have basic ability in using one specific operating system and should be aware of the existence of other operating systems and networks.
- should be aware of the main stages in systems analysis.

- should appreciate the widespread use of ICT worldwide.
- are able to solve simple linear algorithms.
- show familiarity with the simple constructs of the programming language studied and interpret simple programs.
- should have a basic understanding of the practical problems involved when using computers.

GRADE 5

Candidates:

- show average knowledge and understanding of the wide range use of computers today in information processing.
- should show an average ability in using common office application software along with communications software.
- show average knowledge of computer hardware and peripherals.
- show average knowledge of how the various components function and communicate together as a complete system.
- should have average ability in using one specific operating system and should be aware of the existence of other operating systems, networks and their characteristics.
- should have a wider understanding of the main stages in systems analysis.
- should appreciate the widespread use of ICT and its effects on the wider world.
- are able to solve algorithms including decisions.
- show a wider knowledge of simple constructs of the programming language studied including the coding, interpretation and testing of simple programs.
- should have an understanding of the practical problems involved when using computers.

GRADE 1

Candidates:

- show a good knowledge and understanding of the wide-range use of computers today in information processing.
- should show versatility in using common office application software along with communications software.
- show more detailed knowledge of computer hardware and peripherals.
- show a deeper knowledge of how the various components function and communicate together as a complete system.
- show greater ability in using one specific operating system and should be aware of the existence of other operating systems, networks and their characteristics.
- should have a deeper understanding of the main stages in systems analysis and be able to apply them in system development.
- should be able to differentiate amongst and appreciate the widespread use of ICT applications and their effects on the wider world.
- show mastery in solving algorithms including simple low level language problems.
- show a deeper knowledge of the constructs of the programming language studied including the coding, interpretation and testing of more complex programs.
- should have a good understanding of the practical problems involved when using computers.

SYLLABUS

PART 1
COMPUTER APPLICATIONS

SYLLABUS	SUPPLEMENTARY NOTES

OBJECTIVES

The candidate should be aware of the widespread use of computers today in processing information and should have first-hand experience of the use of common office application software.

The primary aim of PART 1 is to introduce the computer system from the **user's** point of view, using **practical** examples. This is intended to:

1. motivate candidates, showing them the power and potential of computer systems, and
2. provide candidates with a first-hand knowledge of various topics to be discussed in greater detail later in the course. The teacher should be able to draw on the practical experience gained by candidates in this part to illustrate points arising in the more theoretical parts of the course.

1.1 THE COMPUTER SYSTEM

The computer system as an information processing machine. Its tasks of handling information: inputting, processing, outputting, storing, retrieving, sending and receiving information.

The teacher is to introduce this section with a demonstration application encompassing word processing, spreadsheet and database, as explained above. Students can then modify an existing spreadsheet and watch the effect on screen, perform a mail-merge using the word processor, etc.

It is required that the teacher design a carefully thought-out demonstration application from everyday life. The application should encompass the most widely-used packages nowadays - spreadsheet, database and word processor. Moreover, it is imperative that the demonstration should emphasise the **integrated** use of these packages, and not merely present each one in isolation.

1.2 SERIAL AND DIRECT METHODS OF ACCESS

Serial and direct access and their suitability of use for certain applications (e.g. serial for payrolls, direct access for airline booking reservations).

Teachers should emphasise that serial and direct are access MODES - some devices are capable of supporting both access modes (e.g. disk), while others can only support serial access (e.g. tape).

1.3 COMMON APPLICATION SOFTWARE

Classical commercial packages – word processor, database management system, and spreadsheet. Other popular software packages – utilities, anti virus software, graphics.

Basic skills in using typical software packages mentioned above. Use of applications software under a windowing environment.

The ability to suggest the most suitable software for use in specific environments.

The ability to compare and contrast different packages.

1.4 THE SPREADSHEET

Its use to process information. Only simple understanding required of, e.g., storage of data in cells as a label, value or formula; simple calculations; copying and moving cells; data graphing; printing.

One demonstration of graphs and one example from profit analysis, costing, budgeting, income-tax calculation or projections is recommended.

1.5 THE WORD PROCESSOR

Its advantages and ease of use to produce a simple document such as a letter. Its main features such as fast, easy entry and editing of text, word wrap, margin justifications, centring of text, underlining, indentations, page layout, blocks, find, find/replace, spell check, mail merge.

Candidates should be given a simple example document to be word processed and a simple model to be worked out using a spreadsheet. Similarly, candidates should be introduced to DTP and a drawing package through practical use of these application packages. Examination questions will concentrate on the features commonly supported by these packages, how they differ from each other (e.g. word processor vs. DTP), and their suitability for a particular application.

Main desktop publishing features of Word Processing such as automatic table of contents and index creation, multi-column documents, tables, frames, embedded graphic objects, etc.

At least one demonstration of the use of these features should be carried out, giving candidates themselves a chance to create more sophisticated documents.

1.6 GRAPHICS PACKAGES

Basic tools to create a simple graphic and its inclusion into text documents.

It is suggested that students be encouraged to modify clipart pictures and/or create their own using a graphics program before transfer to a text document.

1.7 COMMUNICATIONS TOOLS

THE WEB BROWSER

Simple use of a web browser to access web sites and for searching using a popular search engine. Organising sites in folders. Navigating among sites using the web browser.

E-MAIL

Using an e-mail program to send and receive messages. Saving, printing and deleting a message. Advantages and disadvantages of e-mail compared to the postal system.

CREATING WEB PAGES

Using a web authoring program to edit/create a simple web page that includes text, graphics and buttons for linking to other sites.

No working knowledge of HTML languages is required but viewing of the source code is advisable. Web authoring features included in Office applications may also be used.

PART 2
COMPUTER ARCHITECTURE AND DATA REPRESENTATION

SYLLABUS

SUPPLEMENTARY NOTES

OBJECTIVES

The candidate should be familiar with the main hardware functional units of a computer system and how these fit together and communicate to form a complete system.

Because this part of the syllabus tends to be somewhat theoretical, special effort should be made by the teacher to present this material in as concrete a way as possible. Wherever possible, students should be shown the hardware units discussed in this section, and preferably be allowed to use/handle them. Other instructional aids required include sample output from the various output devices discussed, and samples of media (diskette, printer ribbon, inkjet head, etc.). Many excellent videos demonstrating the internal workings of a digital computer in an easy-to-understand and appealing way are currently available.

The features of a computer are frequently mentioned prominently in advertisements and brochures (which are typically intended for the average layman), and for this reason it is felt that candidates should be sufficiently well-informed to make sense of these features.

2.1 DATA REPRESENTATION

2.1.1 NUMBER SYSTEMS

Representation of numbers in binary. The concept of a register. Operations of register, complementation, ranges, left and right shifts. Numerical overflow.

Conversion between decimal and binary, and between binary and hexadecimal.

The aim is to demonstrate how GROUPS of binary signals can be used to represent data which can assume more than just two values. For this reason, it is sufficient to demonstrate the principle using only non-negative integers. It is also important that the candidate understand the relation between the RANGE of values the data may assume, and the number of bits required to represent it. Hence, how numerical overflow may occur when adding or multiplying two numbers. Knowledge of octal is not required. Hexadecimal should be presented ONLY as a shorthand notation for binary.

** 2's complement representation. Binary addition and subtraction (by complementation followed by addition) in 2's compliment.*

Use subscripts for bases: 2, 10 and 16 for binary, decimal and hexadecimal respectively.

** For paper 2A, it is required that the candidate also knows how negative integers may be represented in binary. Although not directly mentioned in the syllabus, it makes sense to first discuss sign-magnitude representation before introducing 2's complement representation. Questions about addition and subtraction will however only be set using 2's complement.*

2.1.2 Coding Systems

Range of possible symbols that can be represented by a given number of bits.

Representation of text using an 8-bit coding system (e.g. ASCII).

Candidates need to appreciate that groups of bits need not only represent numerical data, but can also represent textual data if an appropriate coding system is used. Some awareness of how the choice of coding system affects data portability is also required (e.g. problems which arise when transferring textual data between ASCII and EBCDIC machines).

** Importance of collating sequence in a character-coding system (e.g. the ASCII code). Problems with representing international character sets.*

** Much thought has gone into devising character codes which simplify text processing, for example sorting textual data in alphabetical order, distinguishing upper from lower case characters, etc. Candidates need to be aware of these considerations, as well as of the need for devising a character set which can handle all international languages as global data communication networks become a reality.*

2.1.3 Preparation of Data for Inputting into The Computer System

Data capture forms (very simple), preparation and transcription of data with their related errors; solution through data verification and validation; check digits; range check.

For example, a stock control application can be used to show the connection between analysing the system, preparation of data, using a database to store records of items, using a spreadsheet for calculations of profit or loss and a word processor for mail merge.

Knowledge and structure of one application (e.g. stock control, payroll, lending library, school administration, utility accounting, hospital, billing) is required.

Security of files, privacy of information in files and file generations (grandfather, father, son) are covered in Part 5, but should be briefly mentioned here in the context of the demonstration example.

2.1.4 Data Bases

Data and Information

The organization of data in files: files, records (fixed length and variable length records), fields, items, key field.

The teacher is to use a pre-prepared database for demonstration purposes. It is suggested that this database includes records, fields, field types, field overflow, key fields, etc. and is to be made relevant to students' environment.

Updating of data files, inserting new records, deleting unwanted records, and changing items (fields).

File Reports: retrieval of records to view on computer system or print. Importance of speed of response and file structure (serial or direct access comparisons illustrated by an application). Selecting records under certain conditions (e.g. all customers in a certain area). Display or print chosen fields from selected records or from all records.

Practical work in this section should include making modifications to an existing database (the one used in the demonstration application), and creating a very simple database (e.g., an address book).

Students are expected to create a simple relational database with file specifications and using a table, queries, form view and report generation.

Sorting a file according to some criteria, e.g. alphabetically by name or by town.

Relational database: concept of a table and relationships.

Note: 1-1 and 1-many, exclude many-many.

Master files and transactions files - their everyday use.

2.2 COMPUTER ARCHITECTURE

2.2.1 COMPONENTS OF A COMPUTER SYSTEM

Main components of a computer system. CPU, I/O subsystem, main and backing store. Flow of data and control between the main units using buses.

An overall view of a computer system should be presented, with each unit as a black box. The following sections will fill in the details.

2.2.2 BASIC FEATURES OF AN OPERATING SYSTEM

Running several applications concurrently in different windows, easy to use graphics interface, use of clipboard to exchange text and graphics data between applications in particular the integrated use of spreadsheets, word processing and data bases. Managing of files: copying, deleting, renaming, creating of folders/directories.

In this section the operating system is considered from the end-user's point of view. A more technically detailed treatment is given under section 3.5.

2.2.3 COMPUTER LOGIC

Distinction between analogue and discrete processes and quantities. Conversion of analogue quantities to digital form. Using sampling techniques, use of 2-state electronic devices (logic 0 and logic 1) for reliability.

The distinction between analogue (continuous) and discrete quantities and processes should be explained carefully. Some textbooks give examples which are more confusing then enlightening – these should be avoided. The main objective of this section is to demonstrate how different data types can be represented using just 2 symbols (0,1), and how data thus represented can be symbolically manipulated.

Units of storage:

byte	8 bits	
kilobyte	1KB = 1024 bytes	2^{10} bytes
megabyte	1MB = 1024 kilobytes	2^{20} bytes
gigabyte	1GB = 1024 megabytes	2^{30} bytes
terabyte	1TB = 1024 gigabytes	2^{40} bytes

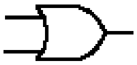
Note: Although there might be different interpretations of the units of storage, for the purpose of this syllabus, the conventions in the table will be used.

2.2.4 LOGIC CIRCUITS

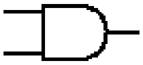
Truth values. OR, AND (2-input only) and NOT gates and their truth tables. Determining the output of a logic circuit containing the three mentioned gates for given inputs.

The simplest use that can be made of a binary signal is in representing truth values (or any other 2-state domain). Logic gates provide the basic mechanism for manipulating data in binary form. Higher-order functions can be implemented by combining gates into circuits. Questions in this section will be strictly confined to the three logical operations AND, OR and NOT.

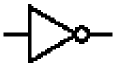
Symbols for Logic Gates:



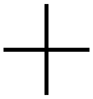
for OR



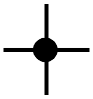
for AND



for NOT



wires not connected



wires connected

Symbols for Boolean Expressions:

A OR B to be represented as ***A + B***

A AND B to be represented as ***A.B***

NOT A to be represented as $\bar{A}$

* *Evaluating a boolean expression (propositional logic formula) - such as* $(\bar{A}+B).C$ *given the values of the boolean variables, by converting to a logic circuit.*

* *Candidates should be able to convert a boolean expression into a logic circuit, and vice versa, and perform simple analysis of such a circuit using truth tables. Candidates are also expected to be able to design a logic circuit to implement a simple boolean function (presented as a propositional expression or as a prose description).*

2.2.5 The CPU

The processor and main memory.

The main components of the processor - control unit, ALU. Registers: accumulator, program counter (PC) and instruction register (IR).

Main memory and memory addresses. The address bus and how its width relates to the size of the address space. The data bus. The computer's word size (or word length). Control bus (read/write line only).

The instruction set as a means of controlling the CPU's circuitry. Function codes (opcodes). Operands. Concept of a machine code program as a set of instructions.

Concept of stored-program. Concept of language levels (machine code being at a lower level than assembly language). Brief account of fetch and execute cycle.

The primary objective of this section is to give the candidate some idea of how the CPU functions - what instructions a processor typically understands, and how these instructions can be held in memory, together with the data to be operated on. No knowledge of assembly language is expected - simply an appreciation of how tedious programming in such a language can be. A demonstration of an assembly language program, together with the contents of the resulting executable, will help students understand the difference between assembly language and machine code.

Note: Only a general account of the fetch execute cycle is required, consisting of the following steps:

1. CU fetches the opcode from memory location indicated by PC
2. CU places opcode in IR
3. CU fetches any required operand
4. CU increments PC to point to next instruction
5. CU activates necessary circuits to execute instruction
6. Go back to step 1

Processor speed-cycles per second expressed in:

Hz	
KHz	10^3 Hz
MHz	10^6 Hz
GHz	10^9 Hz

Units of time measurements: milli, micro, nano seconds.

Frequency units are Hz, KHz (kilo), MHz (mega), GHz (giga). Units of time are seconds, milliseconds, microsecond and nano seconds. It is important that students understand that frequency units and units of time are directly related.

** Typical machine code instructions - load, store and process (e.g. add, sub etc.) instructions. Using mnemonics to represent machine instructions. Immediate, direct and symbolic addressing. Conditional and unconditional branches. Understanding and modifying simple assembly-language programs.*

See Appendix 1.

** Although candidates are not expected to have done any actual assembly-language programming (modern assembly languages are too complex), they should have some paper-and-pencil awareness of how very simple algorithms can be coded in an assembly language. Exposing candidates to some simple assembly language will help them to understand the concepts of language levels more than any amount of verbal description would. Moreover, the operation of a CPU cannot be meaningfully described unless candidates have some knowledge of an assembly language. Questions on this part will concentrate on the candidates' ability to UNDERSTAND and MODIFY a simple snippet of assembly language. Candidates will NOT be required to code algorithms in assembly language.*

2.2.6 STORAGE

Main/primary memory - volatile/writeable (RAM) and non-volatile/non-writeable (ROM), and uses of each (e.g. bootstrap loader in ROM). Secondary/backing storage devices and media - magnetic media (hard disks, tapes, etc), optical media (CD-ROM, DVD-ROM, etc) and electronic media (pendrives). Uses of each, relative capacities and speeds, and access modes (direct vs. sequential). How a disk-drive works - read/write heads, tracks, sectors.

There is no need to differentiate between ROM, PROM and EPROM. However, some mention of battery-backed memory for holding changeable configuration data is in order. The speed differences between main and secondary storage should be emphasised. Care should be taken when explaining the difference between random and sequential access – some candidates get the impression that only a tape drive is capable of sequential access. It is also important that candidates appreciate the SUITABILITY of storage devices for different requirements.

** The disk filing system - storage blocks, disk directory, file allocation. Hierarchical directory structure. Access time (typical, faster or slower only).*

** Candidates should have an idea of how files may be stored on a disk – how a file is allocated a number of sectors, and how a directory keeps track of where each file is stored. For the purpose of this syllabus, it is sufficient to consider only contiguous file allocation. It is also important that candidates are aware of the problem of organising files on high-capacity media, and how hierarchical directory structures can simplify storage organisation.*

2.2.7 I/O DEVICES

INPUT DEVICES

Different types of input devices, and the application areas for which each is best suited. The QWERTY keyboard - alphanumeric and special keys. Pointing devices - the mouse, trackball, trackpoint, touchpad, light pen and touch screen. Bar-code reader, graphics tablet, image scanner and digital camera. Application of input devices such as OMR, MICR and OCR software, handwriting recognition and pen computing.

Candidates should be familiar with the characteristics and typical use of a number of input devices, and appreciate the advantages and disadvantages of each. Where possible, the use of each device discussed should be demonstrated. It is important that the candidate understand that although input devices are not under software control, the interpretation of the signals generated by such devices depends entirely on the software. The candidate is expected to be familiar with at least the following input devices and typical applications:

1. Keyboard (word processing)
2. Mouse (menu selection)
3. Joystick, paddle (game and simulations)
4. Bar-code Reader (point of sale)
5. Magnetic-Ink Reader (bank applications)
6. Optical scanner
7. Graphics Tablet (graphics application)
8. OMR(correction of multiple-choice examination questions)
9. Audio input (microphone)
10.Pen (handheld or very small computers on which it is physically impossible to fit a keyboard)

OUTPUT DEVICES

Distinction between hard-copy and soft-copy, and between vector (plotter) and raster (laser and dot-matrix printers and screen) devices. Resolution of a raster device and how this relates to both print quality and amount of data that needs to be transferred from the computer to the device (and hence speed).

Candidates need to be familiar with the operating characteristics and typical uses of the following output devices.

1.LCD projectors
2. Dot-matrix printer
3. Inkjet printer
4. Laser printer
5. Plotter
6. VDU/Monitor
7. Audio output (loudspeakers)

The difference in output quality can be easily demonstrated by providing students with samples - if applicable printed at different resolutions. Enlargement of a printout may also help to demonstrate the way the image is formed.

** How an image is displayed on a CRT, FPD (Flat Panel Display) or LCD. Pixels and colour depth. Palettes.*

** Candidates should have some idea of the graphics subsystem of a computer. The material should be approached from an informative rather than a technical point of view. Little depth is expected.*

** A brief overview of the I/O Subsystem: the speed difference between the CPU/RAM and I/O devices. I/O buffering. Disk caching. Serial vs. parallel data transfer.*

** Like the graphics system, the I/O system should be approached from an informative point of view. Candidates need to be aware of how buffering reduces the speed difference between devices, and how caching can reduce the need to access slow devices frequently.*

SPECIAL PURPOSE I/O

Note to teacher: This section is intended to make students aware that people with special needs can still use computers through the use of special devices.

Devices for special needs: braille printers and keyboards, special purpose keyboards, eye sensor readers, speech recognition.

2.2.8 THE OPERATING SYSTEM

System software as the layer between the user/application and the hardware. Resource management functions; Management of:

- files
- memory
- CPU
- I/O

Candidate should appreciate the role the Operating System, and system software in general, plays in a computer system. They should be familiar, through practical sessions, with hardware aspects of one operating system - file maintenance, loading and run programs, using system utilities, etc.

The O.S. as a communication medium between the units of the machine and the peripheral devices.

2.2.9 DEDICATED COMPUTER SYSTEMS

Difference between a general-purpose and a dedicated computer. Embedded and process control systems. Computerised appliances as an example of dedicated systems – VCR and optical recorders/players, auto pilot, mobile phones, GPS, etc. Specialised I/O devices (sensors, buttons, LCD). Software for dedicated computer systems.

Candidates need to be aware that computers come in many forms, and that input and output peripherals need not be keyboards and screens but can be as simple as a button or a sensor, an LCD display or even a single LED. Embedded systems controlling common appliances are one form of dedicated computers - they have a CPU, I/O devices, software stored in ROM, and some RAM as workspace.

PART 3
COMPUTER SYSTEMS

SYLLABUS | **SUPPLEMENTARY NOTES**

OBJECTIVES

This section is meant to widen the candidates' view of computers and make them aware that some applications are too complex to be handled by a standalone desktop PC. It is recommended that candidates become familiar with the needs of systems such as airline reservation, police, hospital, finance and banking, weather forecasting and process control of industrial plants.

In section 2.2, candidates familiarised themselves with the use of one specific operating system. In this section, they should be made aware of the existence of other types of operating systems and their characteristics - single-user vs. multi-user, networked systems, single-programming vs. multi-programming. It is strongly recommended that candidates be shown around a large computer installation.

It is important that students understand the need for the different types of operating systems discussed - in other words, that different types of operating systems seek to fulfil different requirements. No knowledge of the internal workings of an O.S. is expected.

3.1 Programming and Application Packages

Off-the-shelf, customisable, and tailor-made packages. Advantages and disadvantages of each.

Software licensing considerations.

** Awareness of the importance of choice of programming language in developing application software. Awareness of 4th generation languages - demonstration of designing a database.*

Students also need to be aware of different user licensing e.g. freeware, shareware, site licensing, upgrades, patches, software registration etc.

The process of installing software and configuration as well as compatibility considerations, technical support and anti-piracy features.

3.2 Roles related to an I.T. Environment

Knowledge of the existence of a wider range of tasks and responsibilities and hence the need to share them. The ability to outline the duties of:

1. Systems Analyst/Designer
2. Programmer
3. I.T. Trainer
4. Data Entry Clerk
5. Web Master
6. Computer technician
7. Computer engineer

Candidates need to appreciate the fact that there is no hard-and-fast rule about how a DP department is organised, and that this depends very much on the size and complexity of the system to be managed. In smaller departments, some of the roles may be the responsibility of a single individual.

3.3 Systems Analysis

Awareness of the process of analysing a system with a view to computerisation:

1. Project selection and feasibility study,
2. Present system study and analysis,
3. Design of new computerised system,
4. Programming and documentation,
5. Implementation and changeover methods,
6. Control and review,
7. System maintenance.

Systems analysis is so far removed from candidates' everyday experience that unless it is introduced using an example system it will make little sense to them. A system should be carefully selected in order to ensure that all (or at least most of) the steps involved in systems analysis can be meaningfully demonstrated. A lending library system is ideal for this exercise, because candidates are familiar with the manual operation of a library.

A stock control example can be used to demonstrate system analysis.

3.4 Networks

Networking: LAN, MAN and WAN. WLAN as a variation of a LAN. The advantages of a LAN over a number of standalone PC's - sharing of hardware and software resources, ease of management/control by the system administrator.

The use of MODEMS to interface computers with telecommunications networks (telephone cable, optic fibre, microwave, satellite links).

Computer communications over WANS; e-mail, WWW, Video Conferencing.

It is important that candidates understand the concept of a resource, and of shareable and non-shareable resources. Ideally, candidates should have had experience with using a LAN (most school labs are equipped with a networked system). This makes it easier for candidates to appreciate many concepts which are otherwise hard to grasp if one's experience is limited to using a standalone computer. The relative advantages and disadvantages of networked and standalone systems should be highlighted. Mention should also be made of distributed databases as an example of sharing non-centralised data resources.

LANs can be connected to form WANs, possibly spanning international borders, using diverse communications means - telephone lines, microwave, satellite etc. Candidates should also understand the importance of a modem to convert digital signals to a form suitable for a transmission medium, and vice-versa.

** Server and client machines; the problem of bandwidth at a general level.*

** General idea that the size of data to be transferred, the time taken for the transfer and bandwidth are directly related*

3.5 Types of Operating Systems

** Real-time, batch, time-sharing (on-line) use. Their meaning - differences shown by examples. The suitability of each operating mode for a particular application.*

** The necessity of different OS's to support modes and configurations mentioned in 3.4.*

** Common types of operating systems – single-user, multi-user, networked, single-programming, multi programming.*

** Consider time critical* ***real-time*** *operating systems as applied in aircraft control systems, nuclear power stations, missile guidance systems, anti-missile attack systems, and other process control systems with a critical response time. Do not include airline booking systems as examples of time critical real-time operating systems.*

Characteristics of real-time system:

a) support application programs which are non-sequential in nature;
b) deal with events occurring concurrently
c) process and produce a response within a guaranteed specified time interval;
d) safely-critical systems with hardware redundancy.

Batch processing *should not be introduced as a historical operating mode, but as a mode which still has its uses and may be the preferred mode of operation under certain circumstances. Batch processing may be easily demonstrated using the operating system shell's script language.*

Awareness of the fact that in large computer systems the OS must be able to manage:

1. a wider range of hardware resources,
2. more users,
3. a greater number of utilities.

It is sufficient for candidates to be able to give examples of these.

PART 4
ALGORITHMIC PROBLEM SOLVING AND PROGRAMMING

SYLLABUS

SUPPLEMENTARY NOTES

OBJECTIVES

The candidate should:

1. understand the task a program is required to perform,
2. design an appropriate algorithm to perform the task,
3. code the algorithm in a high level language,
4. demonstrate an awareness of structured programming techniques,
5. test the program for its performance according to the specifications.

Candidates must have access to a compiler/interpreter for the Java language, as well as to utilities appropriate to the job of editing, compiling, running and debugging a program.

This section should encourage teachers to create a constructive learning environment in which students develop concepts, planning and procedural thinking skills.

- Algorithmic problem solving techniques using the three basic constructs:
  - sequence
  - branching (conditional transfer)
  - iteration (looping)

  concepts should be thoroughly covered before attempting coding.

- A working knowledge Java is required. Questions testing candidates' reading knowledge of a language will be set only in Java.
- Candidates should have practical experience in the form of programming exercises which involve the use of a computer. They should be encouraged to use, criticise and modify a number of programs written by other people.

Java 5 (SDK 1.5 onwards) or later updates should be used.

Suggested IDEs are *JCreator LE* or *BlueJ*.

4.1 DEFINITION AND ANALYSIS OF PROBLEMS

Students should be able to analyse the requirements of a problem and create specifications and target for the solution. Specifications should concentrate on inputs and outputs.

4.2 DESIGNING A SOLUTION TO THE PROBLEM

The development of algorithms.

The use of flow charts for describing an algorithm.

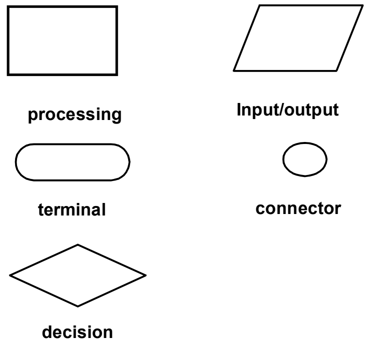
The use of structured programming utilising a top down approach should be emphasised from the beginning.

Pseudo-coding and flowcharting are suitable methods for designing the solution of a problem as we apply the top down method.

Flowchart symbols

Teachers are advised to restrict themselves to a limited number of simple conventional symbols for both program flowcharts and system flowcharts. Program flowcharts should be language independent.

Questions set will be restricted to the following five symbols:



** The use of structure diagrams and pseudo-code for describing an algorithm.*

** Pseudo-code should be used in cases where flowcharts are too cumbersome. The following keywords for developing pseudo-code are suggested:*

***Arithmetic**: add, subtract, multiply, divide, calculate*

***Data transfer**: move, store, replace*

***Input/output**: read, input, output, print*

***Decision**: if…then…else*

***Repetition**: repeat…until, while…endwhile, for…end for.*

4.3 FEATURES OF PROGRAMMING LANGUAGES

Class and method. Data types including strings. Constants and variables, comments, building expressions using arithmetic and assignment operators, operator precedence, input statements (can be done through third party classes that use static methods for keyboard input such as the one provided in the resources of the MATSEC website), output statements, standard class (Math), sequential and conditional execution, looping constructs. Introduction to objects.

Paper setters shall use the Keyboard class when presenting a program listing which contains instructions for the input of data during runtime. However, candidates are free to use the Keyboard class, Scanner class or Console class for their coursework or when asked in the written examination paper/s to develop a Java program or program snippet.

**User-defined methods*

Teaching programming (like teaching any foreign language) should ideally be a gradual process spread over most of the course. The intensive approach to programming does not usually yield good results because programming skills require time to mature and cannot be forced.

Broadly speaking, there are three main stages in learning to program:

- comprehension
- modification
- production.

Not surprisingly, these stages are the same as when we learn a foreign language:

Comprehension – students should first of all learn to understand code. They should be given ready-made programs to analyse through directed experiments where they are required to modify parts of the program and observe the effect this has on program execution. These ready-made programs should also serve to model good coding practices students can later imitate.

Modification – students should next learn to change an existing program in some way: from simple modifications like filling-in the blanks (missing variable declarations, etc.) to more elaborate modifications such as supplying a missing function declaration or extending the program's functionality in some way.

Production – here students are required to produce programs themselves, although even at this stage most students will need lots of scaffolding (such as teacher-supplied libraries or templates).

It is understood that at ALL stages programming is a hands-on exploratory process – at least in this section the computer laboratory should be used in the strict sense of the word: as a place to experiment with code.

The following features should be covered in Java:

(The sequence of the features does not necessarily dictate the order in which they are to be taught).

1. Valid meaningful identifiers, enforcing code convention rules (See Appendix 3), case sensitivity.
2. Variable and constant declarations
3. Data types (primitive), compatibility and type casting
   - byte, short, int, float, char, boolean.
4. The type String can be used for alphanumeric data. No knowledge of String class methods is required.
5. Initialisation and assignment operator - equal sign (=)
6. Comments
   - // (single line) and /* (multi-line)
7. Arithmetic operators, precedence and expressions
   - +, -, *, /, %, ++, --, +=, -=, /= , %=
8. Printing and formatting of text-based output, escape characters:
   - print(), println(), \', \n, \t
     (tutors may use printf () method for formatting purposes especially decimal numbers)
9. Inputting data from the keyboard through a third party class making use of static methods to avoid instantiation of objects just for keyboard input.

10. Logical operators, simple and compound logical expressions
    - !(unary not), && (and), || (or), == (equal to), != (not equal to), > (greater than), >= (greater or equal), < (smaller than), <= (smaller or equal).
11. Loops (including nested loops):
    - while, do while and for
12. Conditional transfer (including at least 2 level nesting):
    - if, if – else, switch
13. Math class – the Math class should be explained as one of the libraries/classes found in the JDK, and limited to the following methods:
    - abs(), pow(), sqrt(), random(), round(), ceil(), floor().
13. Declaration and creation of single-dimensional arrays of primitive data types.
14. Objects – Students create their own classes containing simple void methods and create instances through the use of the keyword 'new'.

** Declaring and calling methods with and without arguments and simple methods which return a value.*

4.4 Testing

Types of errors: syntax, logical, run-time. Candidates should be able to distinguish between them and give examples. Appropriate methods of detection and correction should be understood.

Candidates should be able to identify and correct various types of errors using appropriate diagnostic aids.

Use of suitable test data to check the performance of a program. Output of intermediate results. Dry running. Program tracing.

Candidates should devise and use suitable test data to check that a program performs according to specification, including input of incorrect data by the user.

4.5 Documentation

The difference between user, technical and program documentation.

Candidates should be able to describe and give examples of essential features found in user documentation accompanying software packages.

Candidates will be required to state the items likely to be present in each of the types of documentation and why each of these items is required. Candidates should have access to user manuals of some common application packages.

4.6 Translation of High-Level Languages

Languages of a higher level than assembly language. The need for translators. Source code and executable code. Language translation as a transformation which preserves the semantics of a program - hence a high-level statement may result in many low-level instructions.

Candidates should appreciate how far removed high-level languages are from the machine code instructions understood by the CPU - and how richer high-level constructs are. Hence the one-to-many relation between a high-level construct and the equivalent low-level instructions. This can be demonstrated using natural languages - e.g. certain words in English may translate to whole phrases in Maltese. Samples of high-level source-code in different languages may be used to give students some idea of what programming in such languages is like. Candidates need not completely understand such samples in order to get the idea. It must be made clear that language translators are themselves programs. For the purpose of this syllabus, compilers directly produce executable code - NOT object code (since no mention will be made of program linkers and loaders). High-level language translation will be treated in more detail in Part III, within the context of a specific programming language.

4.7 Language translators

The difference between compilers, interpreters and assemblers - the relative advantages (and disadvantages) of each.

Source code and executable code.

Error messages.

Candidates should be aware of the use of different sets of machine code instructions on different machines and of the common types of operation: arithmetic, logical, fetch and store, branch and input/output. Candidates should be able to understand the simple concepts involved in the functioning of hardware at machine code level.

With regards to assembly language, candidates should appreciate the importance of symbolic addressing for both instructions and data, and the value of mnemonics. The idea of a one-to-one correspondence between an assembly language and a machine code instruction should be understood. It is not expected that candidates have practical experience in programming in an assembly language.

The facilities offered by the various types of translators should be compared. This is best done by means of a demonstration illustrating the basic features such as ease of use, object code, and error messages.

* *4GL's. Software portability.*

** Candidates should have some idea of how 4GL languages differ from 3GL languages, and their advantages (higher-level, automatic code generation, etc.) and disadvantages (size and speed penalty). Although no practical experience of using a 4GL is expected, students should be given a demonstration of some application written using both a 3GL and a 4GL so that they can clearly see the difference. Candidates also need to be aware of the desirability of software portability, and how this relates to language levels.*

PART 5

INFORMATION AND COMMUNICATIONS TECHNOLOGY IN SOCIETY

SYLLABUS | **SUPPLEMENTARY NOTES**

OBJECTIVES

The candidate should:

- be knowledgeable of the major areas of computer applications,
- be aware of the vast amount and variety of computer software available for different applications.
- have an understanding of the practical problems in using computers

Candidates should have access to suitable computer facilities supporting a number of application packages.

5.1 AREAS OF COMPUTER APPLICATIONS

The range of applications familiar to the candidate should include:

- commercial data processing: e.g. stock control, reservations, administration, POS systems.
- technical, mathematical and scientific uses: e.g. medical diagnosis, CAD, simulation, weather forecasting.
- Communication and information systems: Internet and WWW, electronic mail, e-government.

The aim is to give the candidate a knowledge and understanding of a wide range of applications. The general principles and basic concepts are more important that a detailed knowledge of particular equipment.

Excursions to places where students can appreciate the use of computers in different spheres of work and leisure would be highly motivational. Candidates should be familiar with the general concept of forms of applications in particular the following should be emphasised:

global networks as the Internet, and the possibility of browsing this set of networks, using it as a source of information.

• in industry: computer process control: e.g. industrial processes, robotics, gas and oil exploration, monitoring and using energy, CAD-CAM.	CAD-CAM (Computer Aided Design – Computer Aided Manufacturing)
• educational uses: e.g. e-learning, CAL, interschool projects using Internet, school administration.	CAL (Computer Aided Learning); CBT (Computer Based Testing)
• leisure and home uses: e.g. games, microprocessor-controlled home appliances, air-conditioning, security systems.	
• office automation: word processing, database systems, spreadsheets, creation and use of graphics presentation software, personal organizers and schedulers.	
• finances: shops, banks, EFT, stock control, supermarkets, stock exchange, insurance, e-commerce.	EFT (Electronic Fund Transfer)
• travel: air traffic control, navigation (e.g. GPS), 'intelligent' cars, space travel.	
• In the community: police, health, schools, ecological interests, libraries, supermarkets, teleshopping.	

5.2 EFFECTS OF COMPUTER-BASED SYSTEMS ON INDIVIDUALS, ORGANISATIONS AND SOCIETY

Positive and negative effects of computerisation, e.g. satisfaction and efficiency at work, effects of computer games on youngsters, health hazards from working long hours at a computer, opportunity for crime.	Computers are machines which as much as they can be useful, can also have negative effects.

5.3 DATA SECURITY AND PRIVACY

Need of a data security and integrity of data. Backups (the generations of files: grandfather, father, son files), physical security and software safeguards.	Candidates should be aware of methods used for ensuring privacy and integrity of data. The size and usage of computer systems make different backup demands. For example a mainframe computer system of a bank requires different and more sophisticated backup provisions than a small microcomputer system used by a shop owner for stock control purposes.
Malta Data Protection Act - 2001. Students made aware of the Data Protection Act and its provisions and implications for the various sectors and citizens.	The Data Protection Act 2001 - Malta. 1. The Data Protection Act became law in Malta at the end of 2001 and came into effect as from April 2003. 2. It applies equally to public and private sectors 3. Data Controllers regulate compliance with the Data Protection Act especially in Education, Home Affairs, Social Security, Police and Finance.

Important Principles

The Act is about ensuring that Personal Data is:

- Processed fairly and lawfully;
- Always processed in accordance with good practice;
- Collected for specific, explicitly stated and legitimate purposes;
- Processed for reasons compatible with the reason it was collected for;
- Adequate and relevant to the processing purpose;
- Correct and, if necessary, up to date;
- Not kept for longer than it is needed for the processing purpose;
- Completed, corrected, blocked or erased, if the data is found to be incomplete or incorrect with regard to its processing purpose.

Physical security includes both physical facilities to prevent accidental loss such as use of write protect tabs and facilities to deter purposeful corruption and fraudulent use of data, such as restricted access to computer areas. Software facilities include use of IDs and passwords, data encryption, log and audit trail.

Software piracy and copyright. Ethical and legal issues. Hardware and software procedures which deter piracy – serial numbers and activation keys, hardware keys (dongles). Software registration.

Unauthorised copying of software is detrimental for software houses whose existence depends on the sales of their products. This international problem has given rise to laws and hardware and software protection aimed at curbing the crime. Students should be aware of the above problem, security measures and their consequences.

* *Access rights. Privacy on multi-user/network systems.*

5.4 MULTIMEDIA

Brief overview of capabilities and trends. Future perspectives (home office, access to public and institutional databases, libraries, high-quality sound and pictorial data representation). Basic hardware and software requirements and costs.

Candidates are required to have an awareness of this technological development. Though still in experimental stages applications of multimedia are limited only by human imagination. Especially useful for international conferencing and to aid handicapped people.

APPENDIX 1

ASSEMBLY LANGUAGE

The list of limited assembly language instructions given below will be assumed. The operand can either be a memory address (direct addressing), actual data (immediate addressing) or a symbolic address. The symbol # will be used to specify immediate addressing:

```
Eg      ADD  #12      – immediate addressing
        ADD total     – symbolic addressing
        ADD 36        – direct addressing
```

For the purpose of this section the size and type of accumulator is 8-bit unsigned.
Comments are introduced by a semicolon (;).

Data transfer instructions:

```
LDA   x      ; Load accumulator A with x
STA   x      ; Store contents of accumulator A in x
```

Arithmetic, Logical and Shift instructions:

```
ADD   x      ; Add contents of accumulator A with x
SUB   x      ; Subtract contents of x from accumulator A (A minus x)
MUL   x      ; Multiply contents of accumulator A with x
DIV   x      ; Divide contents of accumulator A by x

AND   x      ; Logical AND the contents of accumulator A with x (bitwise operation)
ORA   x      ; Logical OR the contents of accumulator A with x (bitwise operation)
NOT          ; Logical NOT the contents of accumulator A (no operand).

SHL          ; Logical shift contents of accumulator A to the left, introducing a 0 in the
               vacated bit (no operand and bitwise operation).
SHR          ;Logical shift contents of accumulator A to the right, introducing a 0 in the
             vacated bit (no operand and bitwise operation).
```

Transfer of control instructions:

```
JMP   x      ; Jump to the instruction pointed to by the label x (unconditional jump)
JZE   x      ; Jump to the instruction pointed to by the label x if contents of accumulator is 0
             (zero)
JNZ   x      ; Jump to the instruction pointed to by the label x if contents of accumulator is
             not 0 (zero)
```

Other instructions:

```
HLT          ; End of program
```

The operation performed by each instruction is to be provided as a comment in examination papers.

APPENDIX 2

COURSEWORK

The coursework carries a maximum of 30 marks, which account for 15% of the examination assessment. A maximum of 26 marks are allocated for the programming exercise while 4 marks for its layout and presentation. The exercise MUST be word processed.

Teachers are to use the Coursework Marking Scheme sheet available at the end of this appendix, when marking the coursework.

A: PROGRAMMING EXERCISE (maximum mark 26)

This exercise tests the candidate's competence in:

1. Specifying the task which the program is required to perform.
2. Formulating a structured algorithm and describing it by means of a suitable diagrammatic methodology or pseudo-code.
3. Implementing the algorithm in Java using structured techniques.
4. Testing the program using suitable test data.
5. Writing documentation, both program and user instructions.

Notes:

- Candidates are to present only ONE of the programming tasks listed below. However students are free to come up with their own ideas regarding programming tasks, subject to approval by their teacher.
- It is expected that teachers give guidance and suggestions to all candidates in the choice of programming tasks, relating to their ability as well as to hardware and time constraints.
- The high-level language specified in Part 4 (Java) is to be used.
- Candidates opting for either paper 2A or paper 2B are to present a program that makes use of the following:
    - input and output statements
    - assignment expressions
    - sequential execution
    - conditional execution and branching
    - looping construct
    - simple objects

Documentation:

1. Define the task that the program is to perform, including input and output requirements.
2. Provide a description of the algorithm and a hard-copy listing the program.
3. Provide a description of special features and structures used by the program.
4. Provide evidence, in the form of test-run printouts, screen dumps, photos, etc that the program has been fully tested.
5. Provide instructions for a user to run and operate the program
6. Provide an evaluation/criticism of the program.

Criteria for Assessment:

1. Definition of the problem

- A short description indicating the scope of the problem to be tackled. [2]
- A statement of the results required. [2]
- Details of any input information required. [2]

2. The solution of the problem:

- A flowchart/pseudo-code of the algorithm. [4]
- Computer listing of program. [4]
- Details of any special design features. [4]

3. Running the program:

- Evidence that the solution works. [2]
- Details of test data. [2]

4. User instructions:

- Loading and running the program. [2]

5. Comments and conclusion (evaluation of the program as implemented):

- Limitations of the program and possible improvements. [2]

Total Marks: [26]

Guidelines for Awarding Marks:

- Sections with a maximum mark of 2:
  Award marks as follows
  - 0 – not attempted;
  - 1 – partially presented;
  - 2 – fully correct response.
- Algorithm: award the full 4 marks for a complete, readable, understandable, language-independent and easy-to-follow algorithm
- Computer listing: award the full 4 marks for programs showing meaningful identifiers, correct indentation, use of correct coding rules and inclusion of comments
- Special design features: 2 marks are to be awarded for an adequate description of any special features included in the program (good user interface, use of advanced constructs and algorithms, etc). The other 2 marks are to be awarded for a description (and the eventual use) of simple objects.

Examples of Programming Tasks Which May Be Set

- **Class marks** – entry of marks, find the sum, average, lowest, highest, marks above average, below average, grades, simple histogram etc.
- **Payroll** – entry of employee details and fixed rates (Income tax, National Insurance, allowances, deductions), entry of number of hours worked, overtime hours and the program calculates the pay slips showing all the details.
- **Stock Control** – unchanging data can be held in an array and program will produce an invoice, update quantities in stock, search for particular items, and display an inventory list and a suppliers list.
- **Educational Quiz** – An array can be used to hold a question and answer. The program will display the question and the user enters a reply. If the reply is correct then one mark is awarded, if not the correct answer is displayed. A menu may be used to select one topic e.g. a quiz on capital cities, science, sports etc.
- **Telephone directory** – Names and their related telephone numbers (maybe also a town) can be held in an array. User can search for a name and the program will output the telephone number, a town or an address. A menu can be used to output the whole list or searching by a particular town.

- **A mathematical calculator –** using a menu, numbers can be entered and according to choice, the program can perform mathematical calculations or for example find areas of squares, rectangles, circles etc.
- **Hangman** – The program displays underscores according to the number of letters in a word that the user has to guess. If the letter guessed is correct then program displays the letter in its position in the word, if not a part of the Hangman diagram is displayed until the word is guessed or the man is hanged. A number of attempts can be set without the use of drawing the Hangman for simpler programs.
- **Song Contest** – The user enters the judges' total votes and the televoting in two arrays. These are summed up in a third array and the program outputs the winner (the highest). A menu can be used to prompt the user to enter the judges' votes, the televoting, to view each array of votes and to view the winner.
- **A simple game** – Candidates can be creative by moving a simple object on the screen and either try to avoid obstacles or move another object to catch up with the moving object.
- Any **other program** can be developed as long as candidates get their teacher's approval to check that it is at the correct level.

B. LAYOUT AND PRESENTATION (maximum mark 4)

Marks are awarded for an overall good layout and presentation of the programming exercise. This includes clear indications of each section of the programming exercise and the use of word-processing features that help in the clarity of the coursework being presented. Details of this section are available within the Coursework Marking Scheme sheet itself. These features are to be adequately demonstrated where appropriate throughout the coursework.

SECONDARY EDUCATION CERTIFICATE IN COMPUTING

TEACHER –ASSESSED PRACTICAL SCORE

COURSEWORK MARKING SCHEME

IMPORTANT: For moderation purposes by MATSEC, this completed sheet is to be available together with the coursework exercise.

Criteria for Assessment	Maximum Mark	Actual Mark
A: PROGRAMMING		
Definition of the problem:		
Description of the scope of the problem to be tackled	**2**	
Statement of the results required	**2**	
Details of the input information required	**2**	
Solution of the problem:		
Algorithm (flowchart or pseudocode)	**4**	`
Computer listing of program	**4**	
Details of any special design features	**4**	
Running the program		
Evidence that the solution works	**2**	
Plan of test data	**2**	
User instructions:		
Loading and operating the program	**2**	
Comments and conclusion:		
Limitations and improvements	**2**	
TOTAL MAXIMUM FOR PROGRAMMING	**26**	
B: LAYOUT AND PRESENTATION		
Clear indication of each section mentioned above	**1**	
Use of headers and footers (to include page numbers)	**1**	
Use of tables	**1**	
Use of bulleted and/or numbered lists	**1**	
TOTAL MAXIMUM FOR LAYOUT/PRESENTATION	**4**	
TOTAL	**30**	

Guidelines for Awarding Marks:

- Sections with a maximum mark of 2:
  Award marks as follows
  - 0 – not attempted;
  - 1 – partially presented;
  - 2 – fully correct response.
- Algorithm: award the full 4 marks for a complete, readable, understandable, language-independent and easy-to-follow algorithm
- Computer listing: award the full 4 marks for programs showing meaningful identifiers, correct indentation, use of correct coding rules and inclusion of comments
- Special design features: 2 marks are to be awarded for an adequate description of any special features included in the program (good user interface, use of advanced constructs and algorithms, etc). The other 2 marks are to be awarded for a description (and the eventual use) of simple objects.

Student's Name in blocks:

Tutor's Information:

Name in blocks:

Signature:

Date:

APPENDIX 3

Code Convention Rules

The enforcing of code conventions make programs more understandable by making them easier to read. They can also give information about the function of the identifier; for example, whether it is a constant or a class. This can be helpful in understanding the code.

Identifier Type	**Rules for Naming**	**Examples**

Classes	Class names should be nouns in singular, in mixed case with the first letter of each internal word capitalized. Keep class names simple and descriptive.	class Raster;
		class ImageSprite;

Methods	Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.	run();
		runFast();
		selectFromMenu();

Variables	Variable names should be short yet meaningful. The choice of a variable name should indicate to the casual observer the intent of its use. One-character variable names should be avoided.	int num;
		char letter;
		float myWidth;

Constants	The names of variables declared class constants should be all uppercase with words separated by underscores ("_").	final int MIN_WIDTH = 4;
		final int SPEED_LIMIT = 60;

Code block – Opening first curly bracket same line as construct and closing on a separate line.

Example:

```
if (     ) {
} else {
}
```