

## Topic 1—System fundamentals (20 hours)

### 1.1 Systems in organizations (10 hours)

	Assessment statement	Obj	Teacher's notes
<b>Planning and system installation</b>			
1.1.1	Identify the context for which a new system is planned.	2	The extent and limitations of a new system should be appreciated.  Organizational issues related to the installation of new systems such as user roles, underlying technologies.
1.1.2	Describe the need for change management.	2	Students should understand there are a number of factors that need to be managed to ensure change is successful.  <b>S/E</b> The way that change is managed can have significant effects on employers and employees.
1.1.3	Outline compatibility issues resulting from situations including legacy systems or business mergers.	2	<b>INT, S/E</b> When organizations interact, particularly on an international basis, there may be issues of software compatibility and language differences.
1.1.4	Compare the implementation of systems using a client's hardware with hosting systems remotely.	3	The benefits and drawbacks of SaaS (Software-as-a-Service) should be considered.  <b>S/E, INT, AIM 8</b> The remote host may be in a different time zone and this can have significant effects on end-users.
1.1.5	Evaluate alternative installation processes.	3	Students should be aware of the methods of implementation/ conversion.  Parallel running, pilot running, direct changeover and phased conversion.  <b>S/E</b> Training issues may require organizations to restructure their workforce.
1.1.6	Discuss problems that may arise as a part of data migration.	3	<b>INT</b> These include incompatible file formats, data structures, validation rules, incomplete data transfer and international conventions on dates, currencies and character sets.

	Assessment statement	Obj	Teacher's notes
1.1.7	Suggest various types of testing.	3	<p>The crucial importance of testing at all stages of implementation should be emphasized, with the stages clearly defined.</p> <p>Types of testing can include: user acceptance testing, debugging, beta testing.</p> <p>Students should be aware that there are programs that can test other programs, thereby automating parts of the testing process and reducing costs.</p> <p><b>S/E</b> Inadequate testing can reduce employee productivity and lead to end-user dissatisfaction.</p>
<b>User focus</b>			
1.1.8	Describe the importance of user documentation.	2	<b>S/E</b> The quality of user documentation can affect the rate of implementation of the new system.
1.1.9	Evaluate different methods of providing user documentation.	3	<p>Examples should include methods such as: help files, online support and printed manuals.</p> <p><b>S/E</b> The quality of user documentation can affect the rate of implementation of the new system.</p>
1.1.10	Evaluate different methods of delivering user training.	3	<p>Examples should include self-instruction, formal classes, remote/online training.</p> <p><b>S/E</b> The quality of the delivery of user training can affect the rate of implementation of the new system.</p>
<b>System backup</b>			
1.1.11	Identify a range of causes of data loss.	2	<p>Causes include malicious activities and natural disasters.</p> <p><b>S/E</b> Malicious activity may be a result of activities by employees within the organization or intruders.</p>
1.1.12	Outline the consequences of data loss in a specified situation.	2	<b>S/E</b> Loss of medical records, cancellation of a hotel reservation without the knowledge of the traveller.

	Assessment statement	Obj	Teacher's notes
1.1.13	Describe a range of methods that can be used to prevent data loss.	2	These should include failover systems, redundancy, removable media, offsite/online storage.
<b>Software deployment</b>			
1.1.14	Describe strategies for managing releases and updates.	2	<p>Students should be aware of a variety of ways in which updates and patches are made available and deployed. This includes automatic updates received on a regular basis online.</p> <p><b>S/E, INT</b> Performance issues related to the inability to install updates may hinder end-users and reduce compatibility between systems in geographically diverse locations.</p>

## 1.2 System design basics (10 hours)

	Assessment statement	Obj	Teacher's notes
<b>Components of a computer system</b>			
1.2.1	Define the terms: hardware, software, peripheral, network, human resources.	1	
1.2.2	Describe the roles that a computer can take in a networked world.	2	Roles include client, server, email server, DNS server, router and firewall.
1.2.3	Discuss the social and ethical issues associated with a networked world.	3	<b>AIM 8, AIM 9</b> Develop an appreciation of the social and ethical issues associated with continued developments in computer systems.
<b>System design and analysis</b>			
1.2.4	Identify the relevant stakeholders when planning a new system.	2	<p><b>S/E</b> The role of the end-user must be considered when planning a new system.</p> <p>Who is a relevant stakeholder?</p> <p><b>TOK</b> Utilitarianism, the greatest good for the greatest number. The means justify the ends.</p>

	Assessment statement	Obj	Teacher's notes
1.2.5	Describe methods of obtaining requirements from stakeholders.	2	<p>Including surveys, interviews, direct observations.</p> <p><b>AIM 5</b> The need for effective collaboration to obtain appropriate information from stakeholders.</p> <p><b>S/E</b> The question of privacy for stakeholders.</p>
1.2.6	Describe appropriate techniques for gathering the information needed to arrive at a workable solution.	2	<p>Examining current systems, competing products, organizational capabilities, literature searches.</p> <p><b>S/E</b> Intellectual property.</p>
1.2.7	Construct suitable representations to illustrate system requirements.	3	<p>Examples include: system flow charts, data flow diagrams, structure chart.</p> <p>UML is not required.</p> <p><b>LINK</b> Flow chart symbols, flow charts and pseudocode.</p>
1.2.8	Describe the purpose of prototypes to demonstrate the proposed system to the client.	2	<p><b>AIM 5</b> The need to effectively collaborate to gather appropriate information to resolve complex problems.</p> <p><b>AIM 6</b> To develop logical and critical thinking to develop proposed systems.</p>
1.2.9	Discuss the importance of iteration during the design process.	3	<b>MYP</b> Design cycle.
1.2.10	Explain the possible consequences of failing to involve the end-user in the design process.	3	<p><b>S/E</b> The failure to involve the end-user may lead to software that is not suitable for its intended use, which may have adverse effects on user productivity.</p> <p><b>AIM 5</b> The need for effective collaboration and communication between the client, developer and end-user.</p>
1.2.11	Discuss the social and ethical issues associated with the introduction of new IT systems.	3	<b>AIM 8, AIM 9</b> Develop an appreciation of the social and ethical issues associated with continued developments in specified computer systems.

	Assessment statement	Obj	Teacher's notes
<b>Human interaction with the system</b>			
1.2.12	Define the term usability.	1	<b>S/E</b> This includes ergonomics and accessibility.
1.2.13	Identify a range of usability problems with commonly used digital devices.	2	<b>S/E</b> Students should be aware of usability issues in a range of devices including PCs, digital cameras, cell phones, games consoles, MP3 players and other commonly used digital devices.
1.2.14	Identify methods that can be used to improve the accessibility of systems.	2	<b>S/E</b> Examples include touch screen, voice recognition, text-to-speech, Braille keyboard.
1.2.15	Identify a range of usability problems that can occur in a system.	2	<b>S/E</b> These should be related to the systems.  Systems include ticketing, online payroll, scheduling, voice recognition, systems that provide feedback.
1.2.16	Discuss the moral, ethical, social, economic and environmental implications of the interaction between humans and machines.	3	<b>AIM 8</b> Raise awareness of the moral, ethical, social, economic and environmental implications of using science and technology.


## Topic 2—Computer organization (6 hours)

### 2.1 Computer organization (6 hours)

	Assessment statement	Obj	Teacher's notes
<b>Computer architecture</b>			
2.1.1	Outline the architecture of the central processing unit (CPU) and the functions of the arithmetic logic unit (ALU) and the control unit (CU) and the registers within the CPU.	2	Students should be able to reproduce a block diagram showing the relationship between the elements of the CPU, input and output and storage. The memory address register (MAR) and memory data register (MDR) are the only ones that need to be included.
2.1.2	Describe primary memory.	2	Distinguish between random access memory (RAM) and read-only memory (ROM), and their use in primary memory.

	Assessment statement	Obj	Teacher's notes
2.1.3	Explain the use of cache memory.	3	Students should be able to explain the effect of cache memory in speeding up the system as well as being able to explain how it is used.
2.1.4	Explain the machine instruction cycle.	3	This should include the role of data bus and address bus.
<b>Secondary memory</b>			
2.1.5	Identify the need for persistent storage.	2	<p>Persistent storage is needed to store data in a non-volatile device during and after the running of a program.</p> <p><b>LINK</b> Consequences of data loss.</p> <p><b>TOK</b> If there are no consequences of data loss, why is it stored.</p> <p><b>TOK</b> There is no such thing as persistent storage.</p> <p><b>AIM 9</b> An appreciation of the issues related to both the ever increasing amount of data and a need to retain it.</p>
<b>Operating systems and application systems</b>			
2.1.6	Describe the main functions of an operating system.	2	This is confined to a single-user operating system. Technical details are not needed. For example, memory management should be described but how this is handled in a multitasking environment is not expected.
2.1.7	Outline the use of a range of application software.	2	Application software should include word processors, spreadsheets, database management systems, email, web browsers, computer-aided design (CAD) and graphic processing software.

	Assessment statement	Obj	Teacher's notes
2.1.8	Identify common features of applications.	2	<p>Including toolbars, menus, dialogue boxes, graphical user interface (GUI) components.</p> <p>Students should understand that some features are provided by the application software and some by the operating system.</p> <p><b>S/E</b> This improves usability for a wide range of users.</p> <p><b>AIM 9</b> An appreciation of the improvements associated with developments in application software.</p>
<b>Binary representation</b>			
2.1.9	Define the terms: bit, byte, binary, denary/decimal, hexadecimal.	1	
2.1.10	Outline the way in which data is represented in the computer.	2	<p>To include strings, integers, characters and colours. This should include considering the space taken by data, for instance the relation between the hexadecimal representation of colours and the number of colours available.</p> <p><b>TOK, INT</b> Does binary represent an example of a lingua franca?</p> <p><b>S/E, INT</b> Comparing the number of characters needed in the Latin alphabet with those in Arabic and Asian languages to understand the need for Unicode.</p>
<b>Simple logic gates</b>			
2.1.11	Define the Boolean operators: AND, OR, NOT, NAND, NOR and XOR.	1	<b>LINK</b> Introduction to programming, approved notation sheet.
2.1.12	Construct truth tables using the above operators.	3	<p>For example, Maria won't go to school if it is cold and raining or she has not done her homework.</p> <p>Not more than three inputs are used.</p> <p><b>LINK</b> Thinking logically.</p> <p><b>TOK</b> Reason as a way of knowing.</p>

	Assessment statement	Obj	Teacher's notes
2.1.13	Construct a logic diagram using AND, OR, NOT, NAND, NOR and XOR gates.	3	<p>Problems will be limited to an output dependent on no more than three inputs.</p> <p>The gate should be written as a circle with the name of the gate inside it. For example:</p>  <p><b>LINK</b> Thinking logically, connecting computational thinking and program design, introduction to programming.</p>

## Topic 3—Networks (9 hours)

### 3.1 Networks (9 hours)

	Assessment statement	Obj	Teacher's notes
<b>Network fundamentals</b>			
3.1.1	Identify different types of networks.	2	<p>Examples include local area network (LAN), virtual local area network (VLAN), wide area network (WAN), storage area network (SAN), wireless local area network (WLAN), internet, extranet, virtual private network (VPN), personal area network (PAN), peer-to-peer (P2P).</p> <p><b>S/E, INT</b> Globalization has been accelerated by the technical advances linked to network development.</p>
3.1.2	Outline the importance of standards in the construction of networks.	2	<b>INT</b> Standards enable compatibility through a common “language” internationally.
3.1.3	Describe how communication over networks is broken down into different layers.	2	Awareness of the OSI seven layer model is required, but an understanding of the functioning of each layer is not.
3.1.4	Identify the technologies required to provide a VPN.	2	
3.1.5	Evaluate the use of a VPN.	3	<b>S/E, AIM 9</b> The use of a VPN has led to changes in working patterns.



	Assessment statement	Obj	Teacher's notes
<b>Data transmission</b>			
3.1.6	Define the terms: protocol, data packet.	1	
3.1.7	Explain why protocols are necessary.	3	Including data integrity, flow control, deadlock, congestion, error checking.
3.1.8	Explain why the speed of data transmission across a network can vary.	3	
3.1.9	Explain why compression of data is often necessary when transmitting across a network.	3	<b>S/E, INT</b> Compression has enabled information to be disseminated more rapidly.
3.1.10	Outline the characteristics of different transmission media.	2	Characteristics include: speed, reliability, cost and security.  Transmission media include: metal conductor, fibre optic, wireless.
3.1.11	Explain how data is transmitted by packet switching.	3	
<b>Wireless networking</b>			
3.1.12	Outline the advantages and disadvantages of wireless networks.	2	<b>S/E</b> wireless networks have led to changes in working patterns, social activities and raised health issues.
3.1.13	Describe the hardware and software components of a wireless network.	2	
3.1.14	Describe the characteristics of wireless networks.	2	Include: WiFi; Worldwide Interoperability for Microwave Access (WiMAX); 3G mobile; future networks.  <b>S/E, INT</b> Connectivity between different locations.
3.1.15	Describe the different methods of network security.	2	Include encryption types, userID, trusted media access control (MAC) addresses.  <b>S/E</b> Wireless networks have led to concerns about the security of the user's data.
3.1.16	Evaluate the advantages and disadvantages of each method of network security.	3	

## Topic 4—Computational thinking, problem-solving and programming (45 hours)

### 4.1 General principles (10 hours)

This should not be taught as a separate topic but must be incorporated and connected to all sections—especially flow charts, pseudocode and programming in the SL/HL core and abstract data structures (HL extension). It is essential that these elements are not addressed in isolation—they have to be approached as a whole.

The basic ideas and their application should be illustrated with non-computer examples. Each basic idea should then be practised in specific algorithmic contexts using concepts drawn from flow charts, pseudocode and programming. The teacher support material illustrates examples such as the home/locker/knapsack for thinking ahead.

	Assessment statement	Obj	Teacher's notes
<b>Thinking procedurally</b>			
4.1.1	Identify the procedure appropriate to solving a problem.	2	This includes identifying the steps and putting them in the correct order. Such as recipes, block-arrow-block-arrow.  <b>LINK</b> Connecting computational thinking and program design, introduction to programming.
4.1.2	Evaluate whether the order in which activities are undertaken will result in the required outcome.	3	Links to problems presented to the student in other areas of the syllabus.  <b>LINK</b> Thinking ahead, thinking concurrently. Connecting computational thinking and program design, introduction to programming.  <b>MYP</b> Technology, step-by-step instructions.
4.1.3	Explain the role of sub-procedures in solving a problem.	3	Constructing procedures that can then be referred to by their identifier.  <b>LINK</b> Abstraction, connecting computational thinking and program design, introduction to programming.
<b>Thinking logically</b>			
4.1.4	Identify when decision-making is required in a specified situation.	2	Links to procedural thinking—alternative procedures.  <b>TOK</b> Reasoning as a form of decision-making.  <b>LINK</b> Connecting computational thinking and program design, introduction to programming.

	Assessment statement	Obj	Teacher's notes
4.1.5	Identify the decisions required for the solution to a specified problem.	2	Different actions are taken based on conditions.  <b>LINK</b> Connecting computational thinking and program design, introduction to programming.  <b>AIM 4</b> Applying thinking skills to identify and resolve a specified complex problem.
4.1.6	Identify the condition associated with a given decision in a specified problem.	2	Testing conditions, iteration. Identifying and constructing the conditions—AND, OR, NOT relationships—Boolean tests.  <b>LINK</b> Connecting computational thinking and program design, introduction to programming.
4.1.7	Explain the relationship between the decisions and conditions of a system.	3	IF ... THEN ... ELSE  <b>LINK</b> Connecting computational thinking and program design, introduction to programming.
4.1.8	Deduce logical rules for real-world situations.	3	<b>LINK</b> Connecting computational thinking and program design, introduction to programming.
<b>Thinking ahead</b>			
4.1.9	Identify the inputs and outputs required in a solution.	2	
4.1.10	Identify pre-planning in a suggested problem and solution.	2	Gantt charts. Pre-ordering. Pre-heating an oven. Home/locker/knapsack. Caching/pre-fetching. Building libraries of pre-formed elements for future use.  <b>LINK</b> Thinking procedurally, thinking concurrently. Connecting computational thinking and program design, introduction to programming.
4.1.11	Explain the need for pre-conditions when executing an algorithm.	3	
4.1.12	Outline the pre- and post-conditions to a specified problem.	2	For example, cooking a dish for a meal. All ingredients available before starting to cook. A place to eat the food.

	Assessment statement	Obj	Teacher's notes
4.1.13	Identify exceptions that need to be considered in a specified problem solution.	2	For example, identify the pre-conditions for calculating the end-of-year bonus when not all employees have worked for the company for the whole year.  <b>LINK</b> Connecting computational thinking and program design, introduction to programming.
<b>Thinking concurrently</b>			
4.1.14	Identify the parts of a solution that could be implemented concurrently.	2	Could include computer systems or real-life situations.  <b>LINK</b> Thinking ahead, thinking procedurally. Connecting computational thinking and program design, introduction to programming.
4.1.15	Describe how concurrent processing can be used to solve a problem.	2	For example, building a house, production lines, division of labour.  Students will not be expected to construct a flow chart or pseudocode related to concurrent processing.
4.1.16	Evaluate the decision to use concurrent processing in solving a problem.	3	<b>LINK</b> Thinking ahead, thinking procedurally. Connecting computational thinking and program design, introduction to programming.
<b>Thinking abstractly</b>			
4.1.17	Identify examples of abstraction.	2	Selecting the pieces of information that are relevant to solving the problem.  <b>LINK</b> Thinking ahead.
4.1.18	Explain why abstraction is required in the derivation of computational solutions for a specified situation.	3	Students should be aware of the concept of objects, for example, the use of collections as objects in the design of algorithms.  <b>LINK</b> <ul style="list-style-type: none"> <li>Databases: tables, queries</li> <li>Modelling and simulation: an abstraction of reality</li> <li>OOP: classes, sub-classes</li> <li>Web science: distributed applications</li> </ul>

	Assessment statement	Obj	Teacher's notes
4.1.19	Construct an abstraction from a specified situation.	3	There is no need to use code.  Levels of abstraction through successive decomposition.  A school can be decomposed into faculties. A faculty can be decomposed into departments.  <b>LINK</b> Thinking ahead, thinking procedurally. Connecting computational thinking and program design, introduction to programming.
4.1.20	Distinguish between a real-world entity and its abstraction.	2	<b>TOK</b> The map as an abstraction of the territory.

## 4.2 Connecting computational thinking and program design (22 hours)

The focus of this topic is how an understanding of programming languages enhances the students' understanding of computational thinking and provides them with opportunities for practical, hands-on experience of applying computational thinking to practical outcomes.

In externally assessed components questions will be presented using flow charts and/or pseudocode as outlined in the approved notation sheet. Answers will only be required in pseudocode.

Students must be given the opportunity to convert algorithms into working code that is executed and tested.

Working code will not be assessed in the externally assessed components.

	Assessment statement	Obj	Teacher's notes
4.2.1	Describe the characteristics of standard algorithms on linear arrays.	2	These are: sequential search, binary search, bubble sort, selection sort.
4.2.2	Outline the standard operations of collections.	2	These are: addition and retrieval of data.
4.2.3	Discuss an algorithm to solve a specific problem.	3	Students should be expected to discuss the differences between algorithms, including both standard and novel algorithms. For example, discussing the advantages and disadvantages of using a binary search as opposed to a sequential search.

	Assessment statement	Obj	Teacher's notes
4.2.4	Analyse an algorithm presented as a flow chart.	3	<p>Examination questions may involve variables, calculations, simple and nested loops, simple conditionals and multiple or nested conditionals.</p> <p>This would include tracing an algorithm as well as assessing its correctness.</p> <p>Students will not be expected to construct a flow chart to represent an algorithm in an externally assessed component.</p> <p><b>MYP Mathematics:</b> using flow charts to solve problems in real-life contexts, patterns and sequences, logic, algorithms.</p> <p><b>MYP Technology:</b> design cycle (inputs, processes, outputs, feedback, iteration).</p>
4.2.5	Analyse an algorithm presented as pseudocode.	3	<p>Examination questions may involve variables, calculations, simple and nested loops, simple conditionals and multiple or nested conditionals.</p> <p>This would include tracing an algorithm as well as assessing its correctness.</p> <p><b>MYP Mathematics:</b> using flow charts to solve problems in real-life contexts, patterns and sequences, logic, algorithms.</p> <p><b>MYP Technology:</b> design cycle (inputs, processes, outputs, feedback, iteration).</p>
4.2.6	Construct pseudocode to represent an algorithm.	3	<p><b>MYP Mathematics:</b> using flow charts to solve problems in real-life contexts, patterns and sequences, logic, algorithms.</p> <p><b>MYP Technology:</b> design cycle (inputs, processes, outputs, feedback, iteration).</p> <p><b>AIM 4</b> Demonstrate thinking skills to represent a possible solution to a specified complex problem.</p>

	Assessment statement	Obj	Teacher's notes
4.2.7	Suggest suitable algorithms to solve a specific problem.	3	<p>Suitable algorithms may include both standard algorithms and novel algorithms. Suitable may include considerations of efficiency, correctness, reliability and flexibility. Students are expected to suggest algorithms that will actually solve the problem successfully.</p> <p><b>LINK</b> General principles of computational thinking, introduction to programming.</p>
4.2.8	Deduce the efficiency of an algorithm in the context of its use.	3	<p>Students should understand and explain the difference in efficiency between a single loop, nested loops, a loop that ends when a condition is met or questions of similar complexity.</p> <p>Students should also be able to suggest changes in an algorithm that would improve efficiency, for example, using a flag to stop a search immediately when an item is found, rather than continuing the search through the entire list.</p>
4.2.9	Determine the number of times a step in an algorithm will be performed for given input data.	3	Examination questions will involve specific algorithms (in pseudocode/ flow charts), and students may be expected to give an actual number (or range of numbers) of iterations that a step will execute.

### 4.3 introduction to programming (13 hours)

	Assessment statement	Obj	Teacher's notes
<b>Nature of programming languages</b>			
4.3.1	State the fundamental operations of a computer.	1	<p>These include: add, compare, retrieve and store data.</p> <p>Complex capabilities are composed of very large numbers of very simple operations.</p>
4.3.2	Distinguish between fundamental and compound operations of a computer.	2	For example, "find the largest" is a compound operation.

	Assessment statement	Obj	Teacher's notes
4.3.3	Explain the essential features of a computer language.	3	For example, fixed vocabulary, unambiguous meaning, consistent grammar and syntax. <b>TOK</b> Language and meaning.
4.3.4	Explain the need for higher level languages.	3	For example, as the human needs for computer systems have expanded it is necessary to abstract from the basic operations of the computer. It would take far too long to write the type of systems needed today in machine code.
4.3.5	Outline the need for a translation process from a higher level language to machine executable code.	2	For example, compiler, interpreter, virtual machine.
<p><b>Use of programming languages</b></p> <p>Sub-programmes and objects support abstraction, which facilitates: ease of debugging and maintenance, reuse of code, modularity.</p> <p>There is no programming language specified in the SL/HL core. However, students must use a language that supports the basic constructs on the approved notation sheet.</p>			
4.3.6	Define the terms: variable, constant, operator, object.	1	
4.3.7	Define the operators =, ≠, <, <=, >, >=, mod, div.	1	<b>LINK</b> Approved notation sheet.
4.3.8	Analyse the use of variables, constants and operators in algorithms.	3	For example, identify and justify the use of a constant as opposed to a variable in a given situation. <b>MYP</b> Mathematics: forms of numbers, algebra—patterns and sequences, logic, algorithms.
4.3.9	Construct algorithms using loops, branching.	3	Teachers must ensure algorithms use the symbols from the approved notation sheet. <b>LINK</b> Approved notation sheet. <b>MYP</b> Mathematics: using flow charts to solve problems in real-life contexts, logic, algorithms <b>MYP</b> Technology: design cycle (inputs, processes, outputs, feedback, iteration). <b>LINK</b> Connecting computational thinking and program design.



	Assessment statement	Obj	Teacher's notes
4.3.10	Describe the characteristics and applications of a collection.	2	<p>Characteristics:</p> <ul style="list-style-type: none"> <li>Contains similar elements.</li> </ul> <p><b>LINK</b> HL extension, recursive thinking.</p> <p><b>LINK</b> General principles of computational thinking, connecting computational thinking and program design.</p>
4.3.11	Construct algorithms using the access methods of a collection.	3	<b>LINK</b> Connecting computational thinking and program design.
4.3.12	Discuss the need for sub-programmes and collections within programmed solutions.	3	<p>Show an understanding of the usefulness of reusable code and program organization for the individual programmer, team members and future maintenance.</p> <p><b>LINK</b> General principles of computational thinking, connecting computational thinking and program design.</p> <p><b>MYP</b> Technology: use of software such as Alice.</p>
4.3.13	Construct algorithms using pre-defined sub-programmes, one-dimensional arrays and/or collections.	3	<p><b>MYP</b> Mathematics: using flow charts to solve problems in real-life contexts, logic, algorithms.</p> <p><b>MYP</b> Technology: design cycle (inputs, processes, outputs, feedback, iteration); use of software such as Alice.</p> <p>Students will only be required to analyse flow charts in the externally assessed components.</p> <p>Students will be expected to write and analyse pseudocode in the externally assessed components.</p> <p><b>S/E, AIM 8</b> Appreciate the implications of using available code from sources such as online forums.</p> <p><b>LINK</b> Connecting computational thinking and program design.</p>