Topic 1—System fundamentals (20 hours)

1.1 Systems in organizations (10 hours)

	Assessment statement	Obj	Teacher's notes
Planning and system installation			
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. **S/E** 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	**INT**, **S/E** 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. **S/E**, **INT**, **AIM 8** 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. **S/E** Training issues may require organizations to restructure their workforce.
1.1.6	Discuss problems that may arise as a part of data migration.	3	**INT** 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	The crucial importance of testing at all stages of implementation should be emphasized, with the stages clearly defined. Types of testing can include: user acceptance testing, debugging, beta testing. Students should be aware that there are programs that can test other programs, thereby automating parts of the testing process and reducing costs. **S/E** Inadequate testing can reduce employee productivity and lead to end-user dissatisfaction.
User focus			
1.1.8	Describe the importance of user documentation.	2	**S/E** 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	Examples should include methods such as: help files, online support and printed manuals. **S/E** The quality of user documentation can affect the rate of implementation of the new system.
1.1.10	Evaluate different methods of delivering user training.	3	Examples should include self-instruction, formal classes, remote/online training. **S/E** The quality of the delivery of user training can affect the rate of implementation of the new system.
System backup			
1.1.11	Identify a range of causes of data loss.	2	Causes include malicious activities and natural disasters. **S/E** Malicious activity may be a result of activities by employees within the organization or intruders.
1.1.12	Outline the consequences of data loss in a specified situation.	2	**S/E** 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.
Software deployment			
1.1.14	Describe strategies for managing releases and updates.	2	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. **S/E**, **INT** Performance issues related to the inability to install updates may hinder end-users and reduce compatibility between systems in geographically diverse locations.

1.2 System design basics (10 hours)

	Assessment statement	Obj	Teacher's notes
Components of a computer system			
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	**AIM 8**, **AIM 9** Develop an appreciation of the social and ethical issues associated with continued developments in computer systems.
System design and analysis			
1.2.4	Identify the relevant stakeholders when planning a new system.	2	**S/E** The role of the end-user must be considered when planning a new system. Who is a relevant stakeholder? **TOK** Utilitarianism, the greatest good for the greatest number. The means justify the ends.

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

	Assessment statement	Obj	Teacher's notes
Human interaction with the system			
1.2.12	Define the term usability.	1	**S/E** This includes ergonomics and accessibility.
1.2.13	Identify a range of usability problems with commonly used digital devices.	2	**S/E** 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	**S/E** 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	**S/E** 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	**AIM 8** Raise awareness of the moral, ethical, social, economic and environmental implications of using science and technology.

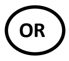
Topic 2—Computer organization (6 hours)

2.1 Computer organization (6 hours)

	Assessment statement	Obj	Teacher's notes
Computer architecture			
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.
Secondary memory			
2.1.5	Identify the need for persistent storage.	2	Persistent storage is needed to store data in a non-volatile device during and after the running of a program. **LINK** Consequences of data loss. **TOK** If there are no consequences of data loss, why is it stored. **TOK** There is no such thing as persistent storage. **AIM 9** An appreciation of the issues related to both the ever increasing amount of data and a need to retain it.
Operating systems and application systems			
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	Including toolbars, menus, dialogue boxes, graphical user interface (GUI) components. Students should understand that some features are provided by the application software and some by the operating system. **S/E** This improves usability for a wide range of users. **AIM 9** An appreciation of the improvements associated with developments in application software.
Binary representation			
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	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. **TOK**, **INT** Does binary represent an example of a lingua franca? **S/E**, **INT** Comparing the number of characters needed in the Latin alphabet with those in Arabic and Asian languages to understand the need for Unicode.
Simple logic gates			
2.1.11	Define the Boolean operators: AND, OR, NOT, NAND, NOR and XOR.	1	**LINK** Introduction to programming, approved notation sheet.
2.1.12	Construct truth tables using the above operators.	3	For example, Maria won't go to school if it is cold and raining or she has not done her homework. Not more than three inputs are used. **LINK** Thinking logically. **TOK** Reason as a way of knowing.

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

Topic 3—Networks (9 hours)

3.1 Networks (9 hours)

	Assessment statement	Obj	Teacher's notes
Network fundamentals			
3.1.1	Identify different types of networks.	2	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). **S/E**, **INT** Globalization has been accelerated by the technical advances linked to network development.
3.1.2	Outline the importance of standards in the construction of networks.	2	**INT** 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	**S/E**, **AIM 9** The use of a VPN has led to changes in working patterns.

	Assessment statement	Obj	Teacher's notes
Data transmission			
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	**S/E**, **INT** 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	
Wireless networking			
3.1.12	Outline the advantages and disadvantages of wireless networks.	2	**S/E** 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. **S/E**, **INT** 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. **S/E** 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
Thinking procedurally			
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. **LINK** 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. **LINK** Thinking ahead, thinking concurrently. Connecting computational thinking and program design, introduction to programming. **MYP** 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. **LINK** Abstraction, connecting computational thinking and program design, introduction to programming.
Thinking logically			
4.1.4	Identify when decision-making is required in a specified situation.	2	Links to procedural thinking—alternative procedures. **TOK** Reasoning as a form of decision-making. **LINK** 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. **LINK** Connecting computational thinking and program design, introduction to programming. **AIM 4** 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. **LINK** 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 **LINK** Connecting computational thinking and program design, introduction to programming.
4.1.8	Deduce logical rules for real-world situations.	3	**LINK** Connecting computational thinking and program design, introduction to programming.
Thinking ahead			
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. **LINK** 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. **LINK** Connecting computational thinking and program design, introduction to programming.
Thinking concurrently			
4.1.14	Identify the parts of a solution that could be implemented concurrently.	2	Could include computer systems or real-life situations. **LINK** 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	**LINK** Thinking ahead, thinking procedurally. Connecting computational thinking and program design, introduction to programming.
Thinking abstractly			
4.1.17	Identify examples of abstraction.	2	Selecting the pieces of information that are relevant to solving the problem. **LINK** 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. **LINK** • Databases: tables, queries • Modelling and simulation: an abstraction of reality • OOP: classes, sub-classes • Web science: distributed applications

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

	Assessment statement	Obj	Teacher's notes
4.2.7	Suggest suitable algorithms to solve a specific problem.	3	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. **LINK** General principles of computational thinking, introduction to programming.
4.2.8	Deduce the efficiency of an algorithm in the context of its use.	3	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. 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.
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
Nature of programming languages			
4.3.1	State the fundamental operations of a computer.	1	These include: add, compare, retrieve and store data. Complex capabilities are composed of very large numbers of very simple operations.
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. **TOK** 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.
Use of programming languages Sub-programmes and objects support abstraction, which facilitates: ease of debugging and maintenance, reuse of code, modularity. 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.			
4.3.6	Define the terms: variable, constant, operator, object.	1	
4.3.7	Define the operators =, ≠, <, <=, >, >=, mod, div.	1	**LINK** 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. **MYP** 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. **LINK** Approved notation sheet. **MYP** Mathematics: using flow charts to solve problems in real-life contexts, logic, algorithms **MYP** Technology: design cycle (inputs, processes, outputs, feedback, iteration). **LINK** Connecting computational thinking and program design.

	Assessment statement	Obj	Teacher's notes
4.3.10	Describe the characteristics and applications of a collection.	2	Characteristics: • Contains similar elements. **LINK** HL extension, recursive thinking. **LINK** General principles of computational thinking, connecting computational thinking and program design.
4.3.11	Construct algorithms using the access methods of a collection.	3	**LINK** Connecting computational thinking and program design.
4.3.12	Discuss the need for sub-programmes and collections within programmed solutions.	3	Show an understanding of the usefulness of reusable code and program organization for the individual programmer, team members and future maintenance. **LINK** General principles of computational thinking, connecting computational thinking and program design. **MYP** Technology: use of software such as Alice.
4.3.13	Construct algorithms using pre-defined sub-programmes, one-dimensional arrays and/or collections.	3	**MYP** Mathematics: using flow charts to solve problems in real-life contexts, logic, algorithms. **MYP** Technology: design cycle (inputs, processes, outputs, feedback, iteration); use of software such as Alice. Students will only be required to analyse flow charts in the externally assessed components. Students will be expected to write and analyse pseudocode in the externally assessed components. **S/E**, **AIM 8** Appreciate the implications of using available code from sources such as online forums. **LINK** Connecting computational thinking and program design.

HL extension (45 hours)

Topic 5—Abstract data structures (23 hours)

5.1 Abstract data structures (23 hours)

This will be examined at the level of diagrams and pseudocode.

Students should be able to describe the most common data structures (arrays, stacks, queues, linked lists, binary trees) and the most common data processing operations on each of the basic data structures (addition, deletion and retrieval of data, traversal, searching for a given data, sorting of data into some order).

This should be taught and connected to flow charts, pseudocode and programming in the SL/HL core.

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.

	Assessment statement	Obj	Teacher's notes
Thinking recursively			
5.1.1	Identify a situation that requires the use of recursive thinking.	2	Suggested practical activity: snowflakes and fractals, towers of Hanoi.
5.1.2	Identify recursive thinking in a specified problem solution.	2	**LINK** Binary trees.
5.1.3	Trace a recursive algorithm to express a solution to a problem.	2	Students will be required to state the output of the recursive algorithm. For example, trees. **LINK** Binary trees.
Abstract data structures			
5.1.4	Describe the characteristics of a two-dimensional array.	2	**LINK** One-dimensional arrays and basic algorithms.
5.1.5	Construct algorithms using two-dimensional arrays.	3	**LINK** Pseudocode information.
5.1.6	Describe the characteristics and applications of a stack.	2	Characteristics: • Last in, first out (LIFO). Examples of the applications of stacks may include running recursive processes, return memory addresses. **LINK** Recursive thinking; connecting computational thinking and program design.

	Assessment statement	Obj	Teacher's notes
5.1.7	Construct algorithms using the access methods of a stack.	3	Access methods: • push • pop • isEmpty. **LINK** Connecting computational thinking and program design.
5.1.8	Describe the characteristics and applications of a queue.	2	Characteristics: • First in, first out (FIFO). Examples of the applications of queues may include print queues and the computer modelling of physical queues (eg supermarket checkouts). Both linear and circular implementation of a queue are required. **LINK** Connecting computational thinking and program design.
5.1.9	Construct algorithms using the access methods of a queue.	3	Access methods: • enqueue • dequeue • isEmpty. **LINK** Connecting computational thinking and program design.
5.1.10	Explain the use of arrays as static stacks and queues.	3	Students should be able to explain push and pop operations, and test on empty/full stack. Students should be able to explain enqueue and dequeue operations, and test on empty/full queue.
Linked lists Linked lists will be examined at the level of diagrams and descriptions. Students are not expected to construct linked list algorithms using pseudocode.			
5.1.11	Describe the features and characteristics of a dynamic data structure.	2	Students should understand the concepts of nodes and pointer.
5.1.12	Describe how linked lists operate logically.	2	**LINK** Logical thinking.

	Assessment statement	Obj	Teacher's notes
5.1.13	Sketch linked lists (single, double and circular).	3	Students should be able to sketch diagrams illustrating: adding a data item to linked list, deleting specified data item, modifying the data held in the linked list, searching for a given data item.
Trees Binary trees will be examined at the level of diagrams and descriptions. Students are not expected to construct tree algorithms using pseudocode. Tracing and constructing algorithms are not expected.			
5.1.14	Describe how trees operate logically (both binary and non-binary).	2	**LINK** Recursive thinking.
5.1.15	Define the terms: parent, left-child, right-child, subtree, root and leaf.	1	These definitions only apply to a binary tree.
5.1.16	State the result of inorder, postorder and preorder tree traversal.	1	
5.1.17	Sketch binary trees.	3	Students should be able to sketch diagrams showing the resulting binary tree after adding a new data item, adding one or more new nodes, and/or removing one or more nodes.
Applications			
5.1.18	Define the term dynamic data structure.	1	
5.1.19	Compare the use of static and dynamic data structures.	3	**LINK** One-dimensional arrays.
5.1.20	Suggest a suitable structure for a given situation.	3	

Topic 6—Resource management (8 hours)

6.1 Resource management (8 hours)

	Assessment statement	Obj	Teacher's notes
System resources			
6.1.1	Identify the resources that need to be managed within a computer system.	2	Resources include: primary memory, secondary storage, processor speed, bandwidth, screen resolution, disk storage, sound processor, graphics processor, cache, network connectivity.

	Assessment statement	Obj	Teacher's notes
6.1.2	Evaluate the resources available in a variety of computer systems.	3	These should include: mainframes, servers, PCs, sub-laptops, as well as personal digital devices such as cell phones, PDAs and digital cameras. **AIM 9** Develop an appreciation of the issues linked to resource availability with continued developments in computer systems.
6.1.3	Identify the limitations of a range of resources in a specified computer system.	2	For example, single processor computers may not be able to render 3D graphics effectively.
6.1.4	Describe the possible problems resulting from the limitations in the resources in a computer system.	2	For example, user time wasted if the primary memory is too small or processor speed inadequate. Multi-access and multi-programming environments should be considered as well as single-user systems.
Role of the operating system			
6.1.5	Explain the role of the operating system in terms of managing memory, peripherals and hardware interfaces.	3	For example, allocating storage and keeping track of programs in memory, swapping between programs on time-slicing, priority or when one is waiting for input.
6.1.7	Outline OS resource management techniques: scheduling, policies, multitasking, virtual memory, paging, interrupt, polling.	2	Technical details as to how these are carried out will not be required, but it is expected that students will be familiar with these techniques and understand when and why they are used.
6.1.8	Discuss the advantages of producing a dedicated operating system for a device.	3	Advantages related to size, speed and customization should be considered. For example, using a dedicated operating system for a cell phone rather than using a pre-existing operating system. **S/E** Issue of proprietary software.

	Assessment statement	Obj	Teacher's notes
6.1.9	Outline how an operating system hides the complexity of the hardware from users and applications.	2	Students should be aware of a range of examples where operating systems virtualize real devices, such as drive letters, virtual memory, input devices, the Java virtual machine. **INT** Issue of localization causing compatibility problems between systems in different countries.

Topic 7—Control (14 hours)

7.1 Control (14 hours)

	Assessment statement	Obj	Teacher's notes
Centralized control systems			
7.1.1	Discuss a range of control systems.	3	A variety of control systems should be examined such as automatic doors, heating systems, taxi meters, elevators, washing machines, process control, device drivers, domestic robots, GPS systems, traffic lights and other common devices. Technical knowledge of specific systems is not expected but students should be able to analyse a specified system. **AIM 9** Develop an appreciation of the possibilities for control systems with developments in computer systems.
7.1.2	Outline the uses of microprocessors and sensor input in control systems.	2	These should be related to the examples suggested above.
7.1.3	Evaluate different input devices for the collection of data in specified situations.	3	Scenarios will be based on familiar situations to students.
7.1.4	Explain the relationship between a sensor, the processor and an output transducer.	3	Technical hardware details are not expected.
7.1.5	Describe the role of feedback in a control system.	2	**LINK** Connecting computational thinking and program design.

	Assessment statement	Obj	Teacher's notes
7.1.6	Discuss the social impacts and ethical considerations associated with the use of embedded systems.	3	**S/E** For example, tagging prisoners, surveillance, CCTV, improved safety systems.
Distributed systems			
7.1.7	Compare a centrally controlled system with a distributed system.	3	Technical hardware details are not expected.
7.1.8	Outline the role of autonomous agents acting within a larger system.	2	Technical hardware details are not expected.

Options (SL 30 hours/HL 45 hours)

Students must take one option.

A—Databases

A.1 Basic concepts (5 hours)

	Assessment statement	Obj	Teacher's notes
A.1.1	Outline the differences between data and information.	2	Data is meaningless. To be useful, data must be interpreted to produce information.
A.1.2	Outline the differences between an information system and a database.	2	Students must be aware that these terms are not synonymous. Databases are a component within an information system. **MYP** Technology.
A.1.3	Discuss the need for databases.	3	This should address topics such as the benefits of data sharing. **S/E** For example, correct information relating to customers and/or clients.
A.1.4	Describe the use of transactions, states and updates to maintain data consistency (and integrity).	2	For example, to ensure data consistency when moving money between two accounts it is necessary to complete two operations (debiting one account and crediting the other). Unless both operations are carried out successfully, the transaction will be rolled back. **S/E** For example, ensuring correct information relating to customers and/or clients.

	Assessment statement	Obj	Teacher's notes
B.4.8	Describe the role of chatbots to simulate conversation.	2	Students should be encouraged to use and analyse such standards as Eliza, Alice and Jabberwacky (as well as any more recent acknowledged ones) to compare conversations. **AIM 9** An appreciation of the possibilities associated with continued developments in computer systems. **MYP** Technology: software such as Alice.
B.4.9	Discuss the latest advances in natural language processing.	3	**AIM 9** An appreciation of the possibilities associated with continued developments in computer systems.

C—Web science

C.1 Creating the web (8 hours)

Students will be expected to have completed practical activities linked to developing different types of web pages and be able to evaluate when a particular type of web page is most appropriate.

	Assessment statement	Obj	Teacher's notes
C.1.1	Distinguish between the internet and World Wide Web (web).	2	
C.1.2	Describe how the web is constantly evolving.	2	Students will be expected to be aware of the major differences between the early forms of the web, Web 2.0, the semantic web and later developments. **MYP** Technology: searching the internet. **AIM 9** Develop an appreciation of the possibilities and limitations associated with the evolution of the web.

	Assessment statement	Obj	Teacher's notes
C.1.3	Identify the characteristics of the following: • hypertext transfer protocol (HTTP) • hypertext transfer protocol secure (HTTPS) • hypertext mark-up language (HTML) • uniform resource locator (URL) • extensible mark-up language (XML) • extensible stylesheet language transformations (XSLT) • JavaScript. • cascading style sheet (CSS).	2	
C.1.4	Identify the characteristics of the following: • uniform resource identifier (URI) • URL.	2	
C.1.5	Describe the purpose of a URL.	2	
C.1.6	Describe how a domain name server functions.	2	
C.1.7	Identify the characteristics of: • internet protocol (IP) • transmission control protocol (TCP) • file transfer protocol (FTP).	2	
C.1.8	Outline the different components of a web page.	2	To include features such as meta-tags, title, etc.
C.1.9	Explain the importance of protocols and standards on the web.	3	**INT** Protocols enable compatibility through a common "language" internationally.
C.1.10	Describe the different types of web page.	2	This should include examples such as personal pages, blogs, search engine pages, forums.
C.1.11	Explain the differences between a static web page and a dynamic web page.	3	To include analysis of static HTML web pages and dynamic web pages, eg PHP, ASP.NET, Java Servlets.
C.1.12	Explain the functions of a browser.	3	

	Assessment statement	Obj	Teacher's notes
C.1.13	Evaluate the use of client-side scripting and server-side scripting in web pages.	3	
C.1.14	Describe how web pages can be connected to underlying data sources.	2	Students will not be expected to write code to indicate how the connection is made, but should understand the principles of connecting to an underlying data source.
C.1.15	Describe the function of the common gateway interface (CGI).	2	
C.1.16	Evaluate the structure of different types of web pages.	3	

C.2 Searching the web (6 hours)

	Assessment statement	Obj	Teacher's notes
C.2.1	Define the term search engine.	1	
C.2.2	Distinguish between the surface web and the deep web.	2	**TOK** Data is always accessible.
C.2.3	Outline the principles of searching algorithms used by search engines.	2	Students will be expected to understand only the principles of the PageRank and HITS algorithms. **LINK** General principles of computational thinking, connecting computational thinking and program design.
C.2.4	Describe how a web crawler functions.	2	Teachers should be aware of the range of terms that can be associated with web crawlers such as bots, web spiders, web robots.
C.2.5	Discuss the relationship between data in a meta-tag and how it is accessed by a web crawler.	3	Students should be aware that this is not always a transitive relationship. **TOK** Data may not always have the intended meaning.
C.2.6	Discuss the use of parallel web crawling.	3	
C.2.7	Outline the purpose of web-indexing in search engines.	2	

	Assessment statement	Obj	Teacher's notes
C.2.8	Suggest how web developers can create pages that appear more prominently in search engine results.	3	Students will be expected to test specific data in a range of search engines, for example examining time taken, number of hits, quality of returns.
C.2.9	Describe the different metrics used by search engines.	2	Students will be expected to test specific data in a range of search engines, for example examining time taken, number of hits, quality of returns. **LINK** General principles of computational thinking. Connecting computational thinking and program design. **S/E**, **AIM 8** An understanding of search engine metrics could lead to exploitation.
C.2.10	Explain why the effectiveness of a search engine is determined by the assumptions made when developing it.	3	Students will be expected to understand that the ability of the search engine to produce the required results is based primarily on the assumptions used when developing the algorithms that underpin it. **LINK** Connecting computational thinking and program design.
C.2.11	Discuss the use of white hat and black hat search engine optimization.	3	**S/E**, **AIM 8** Developers of search engines should have a moral responsibility to produce an objective page ranking.
C.2.12	Outline future challenges to search engines as the web continues to grow.	2	Issues such as error management, lack of quality assurance of information uploaded. **AIM 9** Develop an appreciation that search engines will need to evolve to remain effective as the web grows.

C.3 Distributed approaches to the web (6 hours)

	Assessment statement	Obj	Teacher's notes
C.3.1	Define the terms: mobile computing, ubiquitous computing, peer-2-peer network, grid computing.	1	
C.3.2	Compare the major features of: • mobile computing • ubiquitous computing • peer-2-peer network • grid computing.	3	**LINK** Networks.
C.3.3	Distinguish between interoperability and open standards.	2	
C.3.4	Describe the range of hardware used by distributed networks.	2	Students should be aware of developments in mobile technology that have facilitated the growth of distributed networks.
C.3.5	Explain why distributed systems may act as a catalyst to a greater decentralization of the web.	3	**INT** Decentralization has increased international-mindedness.
C.3.6	Distinguish between lossless and lossy compression.	2	Students will not be required to study the detailed compression algorithms.
C.3.7	Evaluate the use of decompression software in the transfer of information.	3	Students can test different compression methods to evaluate their effectiveness.

C.4 The evolving web (10 hours)

	Assessment statement	Obj	Teacher's notes
C.4.1	Discuss how the web has supported new methods of online interaction such as social networking.	3	Students should be aware of issues linked to the growth of new internet technologies such as Web 2.0 and how they have shaped interactions between different stakeholders of the web. **S/E**, **AIM 8** Emerging technologies are modifying users' behaviour.
C.4.2	Describe how cloud computing is different from a client-server architecture.	2	Student should address the major differences only. **LINK** Networks.

	Assessment statement	Obj	Teacher's notes
C.4.3	Discuss the effects of the use of cloud computing for specified organizations.	3	To include public and private clouds. **S/E**, **AIM 8** Cloud computing could potentially conflict with privacy.
C.4.4	Discuss the management of issues such as copyright and intellectual property on the web.	3	Students should investigate sites such as TurnItIn and Creative Commons.
C.4.5	Describe the interrelationship between privacy, identification and authentication.	2	
C.4.6	Describe the role of network architecture, protocols and standards in the future development of the web.	2	**LINK** Networks. **AIM 9** Develop an appreciation that the future development of the web will have an effect on the rules and structures that support it.
C.4.7	Explain why the web may be creating unregulated monopolies.	3	**INT**, **S/E**, **AIM 8** The web is creating new multinational online oligarchies.
C.4.8	Discuss the effects of a decentralized and democratic web.		**S/E**, **INT** The web has changed users' behaviours and "removed" international boundaries.

HL Extension (15 hours)

C.5 Analysing the web (5 hours)

	Assessment statement	Obj	Teacher's notes
C.5.1	Describe how the web can be represented as a directed graph.	2	The vertices (nodes) represent web pages and the edges represent hyperlinks. It is not a complete graph. The directed graph formed by the web is known as the web graph. **LINK** Mathematics: graph theory.
C.5.2	Outline the difference between the web graph and sub-graphs.	2	A sub-graph will be assumed to be a set of pages linked to one specific topic.
C.5.3	Describe the main features of the web graph such as bowtie structure, strongly connected core (SCC), diameter.	2	Students must be aware the web has a structure that has emerged from the behaviour of web users.

	Assessment statement	Obj	Teacher's notes
C.5.4	Explain the role of graph theory in determining the connectivity of the web.	3	**LINK** Mathematics: graph theory.
C.5.5	Explain that search engines and web crawling use the web graph to access information.	3	Students should be aware of the Page Rank algorithm and explain how it works. No calculations are required.
C.5.6	Discuss whether power laws are appropriate to predict the development of the web.	3	

C.6 The intelligent web (10 hours)

	Assessment statement	Obj	Teacher's notes
C.6.1	Define the term semantic web.	1	
C.6.2	Distinguish between the text-web and the multimedia-web.	2	The traditional web is seen as being text based, the semantic web is multimedia based. **AIM 9** Develop an appreciation of the possibilities and limitations associated with the continuing evolution of the web.
C.6.3	Describe the aims of the semantic web.	2	**S/E**, **AIM 8** Emerging technologies are modifying users' behaviour.
C.6.4	Distinguish between an ontology and folksonomy.	2	
C.6.5	Describe how folksonomies and emergent social structures are changing the web.	2	**S/E**, **AIM 8** Emerging technologies are modifying users' behaviour.
C.6.6	Explain why there needs to be a balance between expressivity and usability on the semantic web.	3	**S/E**, **AIM 8** Emerging technologies are modifying users' behaviour.
C.6.7	Evaluate methods of searching for information on the web.	3	Teachers must address issues relating to searching for non-text based files/multimedia files such as using feature analysis.
C.6.8	Distinguish between ambient intelligence and collective intelligence.	2	

	Assessment statement	Obj	Teacher's notes
C.6.9	Discuss how ambient intelligence can be used to support people.	3	Students will be expected to have researched examples such as biometrics, nanotechnologies. **AIM 9** Develop an appreciation of the possibilities that ambient intelligence provides in supporting people when carrying out routine tasks.
C.6.10	Explain how collective intelligence can be applied to complex issues.	3	Students will be expected to have researched examples such as climate change, social bookmarking and stock market fluctuations. **AIM 5** Engender an awareness that effective collaboration and communication can resolve complex problems. **S/E**, **AIM 8** Emerging technologies are modifying users' behaviour. **TOK** It is possible to have a collective intelligence greater than the sum of the contributors.

D—Object-oriented programming

D.1 Objects as a programming concept (6 hours)

The paradigm of object-oriented programming should be introduced through discussion and example.

	Assessment statement	Obj	Teacher's notes
D.1.1	Outline the general nature of an object.	2	An object as an abstract entity and its components—data and actions. Familiar examples from different domains might be people, cars, fractions, dates and music tracks.
D.1.2	Distinguish between an object (definition, template or class) and instantiation.	2	Students must understand the difference in terms of code definitions, memory use and the potential creation of multiple instantiated objects.
D.1.3	Construct unified modelling language (UML) diagrams to represent object designs.	3	**LINK** Connecting computational thinking and program design.
D.1.4	Interpret UML diagrams.	3	**LINK** Connecting computational thinking and program design.

	Assessment statement	Obj	Teacher's notes
D.1.5	Describe the process of decomposition into several related objects.	2	A simple example with 3–5 objects is suggested. Examples related to D.1.1 could be employers, traffic simulation models, calculators, calendars, media collections. **LINK** Thinking abstractly. **AIM 4** Applying thinking skills critically to decompose scenarios.
D.1.6	Describe the relationships between objects for a given problem.	2	The relationships that should be known are dependency ("uses"), aggregation ("has a") and inheritance ("is a"). **LINK** Thinking abstractly. **AIM 4** Applying thinking skills critically to decompose scenarios.
D.1.7	Outline the need to reduce dependencies between objects in a given problem.	2	Students should understand that dependencies increase maintenance overheads.
D.1.8	Construct related objects for a given problem.	3	In examinations problems will require the students to construct definitions for no more than three objects and to explain their relationships to each other and to any additional classes defined by the examiners. **LINK** Connecting computational thinking and program design. **AIM 4** Applying thinking and algorithmic skills to resolve problems.
D.1.9	Explain the need for different data types to represent data items.	3	The data types will be restricted to integer, real, string and Boolean.
D.1.10	Describe how data items can be passed to and from actions as parameters.	2	Parameters will be restricted to pass-by-value of one of the four types in D.1.6. Actions may return at most one data item.

D.2 Features of OOP (4 hours)

Students should be able to describe the features of OOP that distinguish it from other approaches to computer programming.

	Assessment statement	Obj	Teacher's notes
D.2.1	Define the term encapsulation.	1	Data and actions are limited to the object in which they are defined.
D.2.2	Define the term inheritance.	1	A parent object holds common data and actions for a group of related child objects. Multiple inheritance is not required.
D.2.3	Define the term polymorphism.	1	Actions have the same name but different parameter lists and processes.
D.2.4	Explain the advantages of encapsulation.	3	For example, the scope of data should be confined to the object in which it is defined as far as possible in order to limit side effects and dependencies.
D.2.5	Explain the advantages of inheritance.	3	For example, a parent object holds common data and actions, which enhances reuse and reduces maintenance overheads.
D.2.6	Explain the advantages of polymorphism.	3	For example, an action in a child object may choose to override actions of a parent object. This allows an external program to use the same action on a family of objects without knowing the implementation detail.
D.2.7	Describe the advantages of libraries of objects.	2	For example, sorts and other complex algorithms and processes do not have to be "re-invented".
D.2.8	Describe the disadvantages of OOP.	2	For example, increased complexity for small problems; unsuited to particular classes of problem. **AIM 9** Develop an appreciation of the limitations of OOP.

	Assessment statement	Obj	Teacher's notes
D.2.9	Discuss the use of programming teams.	3	As compared to individuals working alone. Examples include speed to completion, information hiding to reduce module dependencies, expertise in narrow fields (eg testing, documentation), etc. **INT**, **AIM 5** The need to develop a common "language" to enable collaboration across international frontiers when resolving problems.
D.2.10	Explain the advantages of modularity in program development.	3	Advantages include easier debugging and testing, speedier completion, etc.

D.3 Program development (20 hours)

	Assessment statement	Obj	Teacher's notes
D.3.1	Define the terms: class, identifier, primitive, instance variable, parameter variable, local variable.	1	These are generally related to the object's data. See JETS.
D.3.2	Define the terms: method, accessor, mutator, constructor, signature, return value.	1	These are generally related to the object's actions. See JETS.
D.3.3	Define the terms: private, protected, public, extends, static.	1	These are generally related to the OOP features described in D.2. See JETS.
D.3.4	Describe the uses of the primitive data types and the reference class string.	2	In examination questions the primitive types will be limited to int, long, double, char and Boolean. **MYP** Mathematics: forms of numbers.
D.3.5	Construct code to implement assessment statements D.3.1–D.3.4.	3	Students may be asked to trace, explain or construct algorithms using the concepts associated with the terms.
D.3.6	Construct code examples related to selection statements.	3	Students may be asked to trace, explain or construct algorithms using simple and compound if … else constructs.
D.3.7	Construct code examples related to repetition statements.	3	Students may be asked to trace, explain or construct algorithms using for, while or do … while loops.
D.3.8	Construct code examples related to static arrays.	3	Students may be asked to trace, explain or construct algorithms using static arrays.

	Assessment statement	Obj	Teacher's notes
D.3.9	Discuss the features of modern programming languages that enable internationalization.	3	For example, use of UNICODE character sets. **INT** When organizations interact, particularly on an international basis, there may be issues of language differences.
D.3.10	Discuss the ethical and moral obligations of programmers.	3	For example, adequate testing of products to prevent the possibilities of commercial or other damage. Acknowledging the work of other programmers. The main aims of the Open Source movement should be known. **S/E AIM 8** An awareness of the ethical considerations when developing new code.

HL Extension

D.4 Advanced program development (15 hours)

	Assessment statement	Obj	Teacher's notes
D.4.1	Define the term recursion.	1	
D.4.2	Describe the application of recursive algorithms.	2	Students should understand that recursion can be applied to a small subset of programming problems to produce elegant solutions. Students should also understand that recursive algorithms are rarely used in practice. **LINK** Thinking abstractly, thinking recursively.
D.4.3	Construct algorithms that use recursion.	3	This is limited to a method that returns no more than one result and contains either one or two recursive calls. **LINK** Connecting computational thinking and program design.
D.4.4	Trace recursive algorithms.	2	All steps and calls must be shown clearly. **LINK** Connecting computational thinking and program design.
D.4.5	Define the term object reference.	1	As typified by simple classes that are self-referential.
D.4.6	Construct algorithms that use reference mechanisms.	3	

	Assessment statement	Obj	Teacher's notes
D.4.7	Identify the features of the abstract data type (ADT) list.	2	Students should understand the nature of an ADT—where no implementation details are known but the actions/methods are standard.
D.4.8	Describe applications of lists.	2	Students should understand that lists can be used to represent stacks and queues.
D.4.9	Construct algorithms using a static implementation of a list.	3	Lists will be restricted to singly linked types. Methods that should be known are add (head and tail), insert (in order), delete, list, isEmpty, isFull.
D.4.10	Construct list algorithms using object references.	3	Lists will be restricted to singly linked types. Methods that should be known are add (head and tail), insert (in order), delete, list, isEmpty, isFull.
D.4.11	Construct algorithms using the standard library collections included in JETS.	3	The classes are ArrayList and LinkedList. Students should have a broad understanding of the operation of these lists and their interface (methods) but not of the internal structure.
D.4.12	Trace algorithms using the implementations described in assessment statements D.4.9–D.4.11.	2	In examination questions, definitions of ArrayList and LinkedList methods will be given where necessary.
D.4.13	Explain the advantages of using library collections.	3	Students should understand that libraries provide convenient and reliable implementations of common programming tasks.
D.4.14	Outline the features of ADT's stack, queue and binary tree.	2	Students should be able to provide diagrams, applications and descriptions of these ADTs. For example, they should know that a binary tree can be used to efficiently store and retrieve unique keys.
D.4.15	Explain the importance of style and naming conventions in code.	3	Students should understand that meaningful identifiers, proper indentation and adequate comments all improve the readability of code for humans and thus save money, time and effort in programming teams. **INT, AIM 5** The need to develop a common "language" to enable collaboration across international frontiers when resolving problems.