Common Core

Topic 1—Systems life cycle and software development

Students should understand the tasks that a systems analyst would perform when considering a situation that may be computerized. These and subsequent tasks included within the systems life cycle are covered in this topic. An understanding and mastery of some of these aspects is expected to be reflected in the program dossier.

Students should learn to analyse and solve problems, not just to write programs. The software life cycle involves several stages, and students are expected to be involved at some level in all stages. Good systems analysis should include investigation, data collection, careful planning and thorough documentation. If the problem is analysed properly, the implementation will be easier and more successful.

1.1—The systems life cycle 8 hrs

	Assessment statement	Teaching note	Obj
1.1.1	Outline the systems life cycle in terms of the stages: analysis, design, implementation, operation and maintenance.	Other models are acceptable as long as they emphasize the cyclical nature of the problem-solving process.	**2**
1.1.2	Explain the importance of collecting data during the analysis stage.		**3**
1.1.3	Compare methods of data collection.	Examples are: interviewing current users and domain experts, constructing questionnaires, observing current systems and studying prospective user-based documentation.	**3**
1.1.4	Describe the production of a requirements specification during the analysis stage.	This may include: a definition of inputs and outputs, a list of tools, facilities, people available for developing the solution, and a schedule for the next stages of the project.	**2**
1.1.5	Outline the features of a feasibility report.	A feasibility report may be produced during the analysis phase, the design phase, or both. It may include: a brief description of the proposed system; estimated costs; economic, technical and legal responsibility; and a possible completion date.	**2**

© International Baccalaureate Organization 2004

Topic 1—Systems life cycle and software development (continued)

	Assessment statement	Teaching note	Obj
1.1.6	Compare the advantages and disadvantages of alternative solutions in the design stage. This should include both hardware and software solutions.	Several possible solutions should be considered and evaluated by asking questions such as: What kind of output should be produced? Where will the input data come from and how will it be entered? Should the system be centralized or networked? Should computers be used at all? Should standard software packages be used? How much customization is desirable? The emphasis should be on modular organization. The human–computer interface should be considered.	**3**
1.1.7	Discuss methods of testing systems, the importance of proper testing and the implications of inadequate testing.	Students must be able to propose suitable test data, with reasons, during the design and implementation stages.	**3**
1.1.8	Outline methods of implementing new systems.	Methods include: running old systems in parallel with new systems, direct changeover and phased introduction. Training implications and possible disruption during installation should be considered.	**2**
1.1.9	Outline the features and importance of maintaining systems.	Features to be considered include periodic reviews, performance evaluation and clear documentation, to facilitate modifications.	**2**

© International Baccalaureate Organization 2004

Topic 1—Systems life cycle and software development (continued)

1.2—Systems analysis 4 hrs

Students should learn to investigate and analyse problems at the system level before beginning to think about a solution (algorithms). Students should be able to read and construct system flowcharts.

	Assessment statement	Teaching note	Obj
1.2.1	Explain the importance of formulating a problem precisely.		**3**
1.2.2	Discuss the aspects that must be considered in a specified problem.	Students should realize that activities such as interviews, questionnaires and literature searches are required to discover the relevant aspects.	**3**
1.2.3	Identify the outcomes that an appropriate solution must produce to solve a specified problem.		**2**
1.2.4	Identify the parts of a problem that can be solved appropriately using a computer.		**2**
1.2.5	Identify the three basic control constructs of programming: accepting data, processing it and outputting the results.		**2**
1.2.6	Analyse a problem by decomposing it into modules.	For example, the modules might represent the input, process and output of the solution to the problem.	**3**

1.3—Systems design 4 hrs

	Assessment statement	Teaching note	Obj
1.3.1	State the parts of a system.		**1**
1.3.2	Identify the data that needs to be held and processed by a system.		**2**

© International Baccalaureate Organization 2004

Topic 1—Systems life cycle and software development (continued)

	Assessment statement	Teaching note	Obj
1.3.3	Outline suitable means of data capture and output presentation for a system.		**2**
1.3.4	Design appropriate data structures to store data within a system.		**4**
1.3.5	State the hardware components that are appropriate for a system.		**1**
1.3.6	Outline a suitable interface between a system and users.		**2**
1.3.7	Analyse a systems flowchart that represents a complete system.		**3**
1.3.8	Construct a systems flowchart to represent a complete system.	Symbols that students should use appear in appendix 3.	**4**

1.4—Social significance and implications of computer systems 5 hrs

	Assessment statement	Teaching note	Obj
1.4.1	Discuss the social and economic implications of the installation of new systems.	See 1.1.8 for installation methods to be considered.	**3**
1.4.2	Discuss the social significance and implications of the widespread use of computers in society.	The social significance must be treated by reference to economic, political, cultural and environmental consequences. These include: effects on employment (resulting in changes to the working environment, retraining and so on); computers (hacking, viruses and so on); ethical and legal requirements; data storage (preserving privacy, data protection and so on); software users (copyright, software licensing and so on).	**3**
1.4.3	Discuss current trends in computer systems and the consequences of these trends.		**3**

© International Baccalaureate Organization 2004

Topic 1—Systems life cycle and software development (continued)

1.5—Software life cycle 2 hrs

	Assessment statement	Teaching note	Obj
1.5.1	Outline the major stages in the software life cycle.	One model includes: systems analysis, leading to a precise statement of the problem that needs solving (a requirements specification); software design; program construction, including testing and debugging; installation and operation; and maintenance. Other models are acceptable, as long as they emphasize the cyclical nature of the life cycle.	**2**
1.5.2	Explain why software production is normally cyclical.	Students should understand that computer systems are used over long periods of time. The software in these systems requires periodic improvement. After the original design and implementation, further analysis, redesign and restructuring are required to accommodate changing needs. This will continue through many cycles of analysis, design, implementation and use.	**3**

1.6—Software design 8 hrs

	Assessment statement	Teaching note	Obj
1.6.1	Outline the data required to solve a problem that the students have not encountered before, including data file formats, input and output requirements with appropriate user interfaces. For example, screens, OMR forms, report formats.		**2**
1.6.2	Discuss the advantages of modularity in designing a solution to problems.		**3**
1.6.3	Define the term prototyping.		**1**

© International Baccalaureate Organization 2004

Topic 1—Systems life cycle and software development (continued)

	Assessment statement	Teaching note	Obj
1.6.4	Outline the prototyping approach to systems design and development.	Prototyping can be done at many levels of sophistication. For the purposes of this course prototyping is limited to the presentation of a preliminary solution that may not be functional.	**2**
1.6.5	Discuss the advantages to end-users and systems designers of using the prototyping approach.	Prototyping can be used with end-users for the purpose of obtaining feedback at an early stage in the design process. Prototyping can be used by systems designers to investigate alternative solutions to a problem.	**3**
1.6.6	Outline the efficiency of a solution in terms of storage requirements, memory requirements, and speed.	Only a qualitative treatment or a specific calculation is expected; "O" or BigO notation is required only at HL. (See 5.6, Algorithm evaluation.)	**2**
1.6.7	Outline how programs can be tested and debugged.	Testing implies tracing sections of an algorithm, including responses to errors ("dry runs"), as well as the design of test cases, which are then executed. Students must be able to propose suitable test data, and give reasons. Debugging has the components of detecting, diagnosing and correcting errors shown up by testing.	**2**
1.6.8	Describe the role of tools in constructing, testing and debugging programs.	Ideally, students should use an integrated development environment (IDE) combining an editor, interpreter or compiler and debugging tools, but this is not a requirement.	**2**

© International Baccalaureate Organization 2004

Topic 1—Systems life cycle and software development (continued)

1.7—Documentation 4 hrs

	Assessment statement	Teaching note	Obj
1.7.1	Outline why documentation is needed at each stage of the systems life cycle.		**2**
1.7.2	Explain the features of documentation for design, programming and maintenance, that is, system documentation.	Students are required to document their problem-solving process according to the standards described in the guidelines for program dossiers. Program listings must also be thoroughly documented.	**3**
1.7.3	Explain the features of documentation for the user, that is, user documentation.	Students are required to write end-user instructions according to the standards described in the guidelines for the program dossier. Students should know that other user manuals may be needed (for example, on-line help systems and installation manuals where systems are installed by personnel other than end-users), but they are not required to write such documentation.	**3**

© International Baccalaureate Organization 2004

Topic 2—Program construction in Java

2.1—Program construction in Java 50 hrs

Discussion of the material in this subtopic will play a major role in the development of program dossiers. While 50 hours have been allocated it should be noted that some of the 25 hours allocated as teacher contact time will also be used in discussion of these aspects. The high-level language must be Java syntax as specified in appendix 2.

	Assessment statement	Teaching note	Obj
2.1.1	Apply the following high-level language constructs appropriately in order to implement a software design expressed in Java. • Declare variables and types with appropriate scope, distinguishing between private and public identifiers. • Define and apply user-defined objects. • Format output in a user-friendly manner. • Construct and calculate arithmetic, relational and Boolean expressions (only `and`, `or`, `not`) using appropriate operators (&&, || and !) and taking into account their precedence. • Construct and calculate the value of modulo arithmetical expressions “mod”, “div” using appropriate operators (%, /) and taking into account their precedence. • Implement the remaining algorithm constructs in Java: arrays, objects, selection constructs (branching), file operations, iteration constructs (looping), sentinels and flags. • Use built-in subprograms, including those of the Java foundation classes specified in appendix 2. • Define and apply user-defined methods. • Demonstrate an understanding of method signatures. • Demonstrate an understanding of the use of parameters, including object and primitive parameter passing and return values. • Demonstrate an understanding of the scope of Java identities, restricted to the keywords private and public. • Define primitive, class, object, data member, method, method signature and constructor.		**3**

© International Baccalaureate Organization 2004

Topic 2—Program construction in Java (continued)

	Assessment statement	Teaching note	Obj
2.1.2	Apply appropriate data types and data structures to solve a problem that students have not encountered before.	Required data types are integer, real, character and Boolean. Required data structures are strings, one-dimensional arrays, two-dimensional arrays, records and files.	**3**
2.1.3	Describe the nature and function of the data types and data structures stated in 2.1.2.		**2**
2.1.4	Trace algorithms in Java.	See appendix 2. Examination questions will always use Java whenever code needs to be displayed; therefore students must be able to understand algorithms presented in this language. The algorithms may be either the standard algorithms in the syllabus, or algorithms of equivalent complexity that the students have not seen before. The algorithms may use any of the data types and structures listed in 2.1.2.	**2**
2.1.5	Evaluate algorithms written in Java with respect to efficiency, correctness and appropriateness for a task.	See note in 2.1.4.	**3**
2.1.6	Construct algorithms in Java.	See note in 2.1.4.	**4**
2.1.7	Explain the need for searching and sorting.		**3**
2.1.8	Apply sequential (linear) and binary search algorithms, selection and bubble sort algorithms to problems, including some not encountered before.	Searching and sorting provide good examples for studying the design, development and analysis of algorithms. Students should be able to discuss the appropriate circumstances for the use of each algorithm. In examinations they may be given descriptions of other algorithms to be developed.	**3**
2.1.9	Compare the efficiency of the specific searching and sorting algorithms mentioned in 2.1.8.		**3**
2.1.10	Discuss the efficiency of specific searching and sorting algorithms.	BigO notation is not required at SL.	**3**

© International Baccalaureate Organization 2004

Topic 2—Program construction in Java (continued)

	Assessment statement	Teaching note	Obj
2.1.11	Describe syntax errors, logic errors, and run-time errors.	Overflow, underflow and truncation errors may arise during program development and so they may be discussed, but they will not be examined at SL.	**2**

© International Baccalaureate Organization 2004

Topic 3—Computing system fundamentals

This topic covers computer systems (their hardware and software) and how they interact.

3.1—Language translators 2 hrs

	Assessment statement	Teaching note	Obj
3.1.1	Define syntax and semantics.		**1**
3.1.2	Describe the function of high-level language translators.	The translators should be limited to interpreters and compilers.	**2**
3.1.3	Outline the use of software development tools.	Examples include: database management systems, macros, CASE tools and simple language translators (interpreters and compilers are not suitable examples in this context), HTML editor, web page editor, code editor, visual IDE.	**2**

3.2—Computer architecture 12 hrs

	Assessment statement	Teaching note	Obj
3.2.1	Outline the structure of the central processing unit (CPU) including the functions of the control unit (CU), the arithmetic and logic unit (ALU), primary memory and address buses.	Students are expected to be able to reproduce a basic diagram illustrating the CPU and to know that each location in primary memory has a unique address.	**2**
3.2.2	Outline the meaning of the terms bit (b) and byte (B) and their derivatives.	Students must understand that everything in a computer is held and processed in binary, hence the relation between bits, bytes and so on in powers of 2. For example 1kilobyte = 2^{10}. They should be familiar with the prefixes T, G, M, k and their use in computer science measure. They must be able to apply the prefixes T, G, M and k to bits and bytes. For example TB (terabytes), Gb (gigabits), MB (megabytes).	**2**
3.2.3	Outline the meaning of the terms word, register and address and their use in the storage of data and instructions.	The study of specific registers is not required.	**2**

© International Baccalaureate Organization 2004

Topic 3—Computing system fundamentals (continued)

	Assessment statement	Teaching note	Obj
3.2.4	Outline the steps in the machine instruction cycle: fetch, decode, execute and store.	A single processor model is sufficient. The study of a specific CPU is not required.	**2**
3.2.5	Outline the characteristics of primary memory and the difference between volatile and non-volatile memory.	Students must understand the function of RAM, ROM and cache memory and their typical sizes (in bytes). The way in which virtual memory can be used to expand primary memory must be understood but details of paging are not needed.	**2**
3.2.6	Outline the characteristics of secondary memory and define sequential and direct access.	Secondary memory should refer to flash memory, disks, CDs and DVDs and tape. Students must know the type of access of the above secondary memory media. They should also be able to give an application of each type and justify its use for this application.	**2**
3.2.7	Outline the function of a microprocessor designed to perform one or a limited number of functions (within a car, washing machine and so on).	The need for different types of memory in a microprocessor must be understood. Students must be able to quote at least one example of the use of a microprocessor and state the inputs and outputs.	**2**
3.2.8	Discuss the features, advantages, disadvantages and applications of specific input and output devices and the media used by each.	Students must know the following features: mouse, keyboard, touch screen, optical character recognition (OCR), magnetic ink recognition (MICR), scanners (page, mark sense and barcode), LCD panels, speech recognition, sensors, digital cameras, graphics tablets, printers, plotters, monitors, robotics, sound. Technical details are not required unless introduced in the case study.	**3**
3.2.9	Outline recent developments in computer system architecture including processor architecture, primary memory technologies and secondary memory devices.	Technical details are not required unless introduced in the case study.	**2**

© International Baccalaureate Organization 2004

Topic 3—Computing system fundamentals (continued)

3.3—Computer systems 5 hrs

	Assessment statement	Teaching note	Obj
3.3.1	Define the term "operating system".	Knowledge of specific operating systems is not required.	**1**
3.3.2	Outline the functions of operating systems.	Functions include: communicating with peripherals; coordinating concurrent processing of jobs; memory management, resource monitoring, accounting and security; program and data management; providing appropriate user interfaces.	**2**
3.3.3	Discuss the characteristics of various computer systems including single users and multi-users, in both single-tasking and multi-tasking environments.	The terms multi-access and multi programming should be understood but details of the way in which they are managed will not be examined.	**3**
3.3.4	Compare the characteristics and applications of different kinds of computers.	Personal computers, portable computers, mainframes and supercomputers should be considered. Characteristics must include: primary and secondary memory size; input/ouput (I/O) devices; environment (size, convenience, where it is used); cost, users (multi- or single-); and processor (word length, bus size and frequency).	**3**
3.3.5	Outline the principal characteristics of batch processing, online (interactive) processing and real-time processing.		**2**
3.3.6	Outline applications that use each of the processing methods in 3.3.5: batch processing (payroll and bank cheque processing); interactive (online) processing; word processing; computer games; real-time processing (air traffic control and monitoring of patients in hospital intensive care).		**2**
3.3.7	Explain the relationship between master and transaction files.	This should relate to the examples in 3.3.6.	**3**
3.3.8	Discuss the reliability of the system including the implications of failure.	The need for, and use of, backing-up strategies, mirrored systems and the utilities in 3.7.	**3**

© International Baccalaureate Organization 2004

Topic 3—Computing system fundamentals (continued)

3.4—Networked computer systems 8 hrs

	Assessment statement	Teaching note	Obj
3.4.1	Define local area network (LAN), wide area network (WAN), server and client.		**1**
3.4.2	Explain basic network topologies.	Students must be able to explain and illustrate star and bus networks as well as hybrids involving both these networks.	**3**
3.4.3	Explain the hardware required in networking.	Hardware should include communications links (cables, microwave, fibre optics and so on) hub, switch, node and router.	**3**
3.4.4	Define the terms “standard protocol”, “data integrity” and “data security” in the context of data transmission across a network.	Students must know that standard protocols are a set of rules that are internationally recognized in the transmission of data. The difference between data security and data integrity must also be recognized. Students do not need to know specific or technical details such as the ISO (OSI) system of layers, TCP/IP and so on.	**1**
3.4.5	Explain the software involved in networking.	Students must understand the role of communications software in connecting local and wide area networks and the need to deal with protocols and data security.	**3**
3.4.6	Describe suitable methods to ensure data integrity in the transmission of data.	Error-checking codes such as check sums (block character checks) and parity checks must be understood. The reasons for re-transmission should be understood. The quality of communication lines should be considered.	**2**
3.4.7	Describe suitable methods to ensure data security.	Students should understand the concept of data encryption but do not need to give algorithmic details. They must understand the need for, and use of, passwords, physical security and different levels of access (permissions) for different users.	**2**

© International Baccalaureate Organization 2004

Topic 3—Computing system fundamentals (continued)

	Assessment statement	Teaching note	Obj
3.4.8	Discuss the need for speed in data transmission, and how speed can be enhanced.	Students must know that documents and graphics files can be sent in different formats and the format affects the speed of transmission. Common formats such as JPEG and BMP should be known. The principles of data compression should be considered but details of methods are not required.	**3**
3.4.9	Discuss networking applications and the implications of networking for organizations, including internal communications, electronic mail, e-commerce, conferencing and distributed processing.	The use of LANs, public and private WANs and the Internet should be considered.	**3**
3.1.10	Outline the functions of a web browser and search engine including displaying an HTML page, following hyperlinks and searching on key words.	Specific names of browsers and search engines are not required.	**2**

3.5—Data representation 6 hrs

	Assessment statement	Teaching note	Obj
3.5.1	Outline the use of binary to represent data.	Students must understand the relationship between number of digits and number of patterns available (2^n, for example: 4-bit colour representation allows 16 colours; a 32-bit address bus can address 4GB RAM). The different features of ASCII and Unicode should be known but students are not expected to know the specific representations of characters.	**2**
3.5.2	Outline the need for standard formats for storing documents and files.	Link with 3.4.8 and 3.4.9.	**2**
3.5.3	Express numbers in the bases: decimal, binary and hexadecimal.		**2**
3.5.4	Convert integers between the bases specified in 3.5.3 (maximum 8 bits).		**2**

© International Baccalaureate Organization 2004

Topic 3—Computing system fundamentals (continued)

	Assessment statement	Teaching note	Obj
3.5.5	Apply binary notation to represent integers, both positive and negative, using the method-of-two's complement.		**2**
3.5.6	Define analogue data and digital data.		**1**
3.5.7	Outline the need for the interconversion of data between analogue and digital formats for computer processing.	Students need to understand the need for conversion between data for processing, for example, sensors and modems.	**2**
3.5.8	Discuss two applications that require conversion of data between analogue and digital formats including temperature sensing.	Teachers are free to choose the second application. Other software examples include: speech recognition, light detection, image processing, OCR software.	**3**

3.6—Errors 2 hrs

	Assessment statement	Teaching note	Obj
3.6.1	Describe the following causes of errors with reference to an application in each case: data entry, accidental, deliberate, software and hardware.		**2**
3.6.2	Outline methods of detection and prevention for each of the errors in 3.6.1.	Verification and validation should be understood. Check digits and hash totals should be explained. The use of modulo operators (mod, div) in constructing check digits should be understood.	**2**
3.6.3	Describe methods of recovery from an error.	Re-input, re-transmission and restoring from backups should be considered. Error-correcting algorithms are not required.	**2**

© International Baccalaureate Organization 2004

Topic 3—Computing system fundamentals (continued)

3.7—Utility software 2 hrs

	Assessment statement	Teaching note	Obj
3.7.1	Outline the main function(s) of the following software utilities: data compressors, virus software, file managers, defragmentation software.	The required file manager functions are: copy, delete, format, find, create folder/directory, archive, print, back-up, rename and restore. The fact that files are not stored contiguously only needs to be dealt with in outline, in order to understand why defragmentation software is required. Technical details are not required.	**2**
3.7.2	Discuss the need for each of the utilities in 3.7.1.		**3**

© International Baccalaureate Organization 2004

Additional HL material

Topic 4—Computer mathematics and logic

Computer science is not a mathematics course. However, the following topics allow students to understand the basic principles of computer architecture, to understand the fundamental causes of many common errors, to design simple circuits, and to construct some common algorithms requiring mathematical techniques.

4.1—Number systems and representations 6 hrs

	Assessment statement	Teaching note	Obj
4.1.1	Calculate in the bases specified in 3.5.3.	For binary and hexadecimal calculations, only addition is required.	**3**
4.1.2	State the mantissa and exponent of a binary number in floating-point representation. Relate this to scientific notation in decimal.	For negative binary numbers in integer and real formats, only the method-of-two's complement is required.	**1**
4.1.3	Apply binary notation to represent real numbers.	Both fixed-point and floating-point representations are required. Students should be able to calculate the range of normalized floating-point numbers given a specific representation. Issues such as the need for normalization and the loss of precision should be understood.	**2**
4.1.4	Discuss the advantages and disadvantages of integer and floating-point representations.		**3**
4.1.5	Define truncation error, underflow error and overflow error.		**1**
4.1.6	Outline three situations, with each one providing an example of when and where one of the errors in 4.1.5 can occur. Each situation should show a different error, that is all three errors should be described.		**2**

© International Baccalaureate Organization 2004

Topic 4—Computer mathematics and logic

4.2—Boolean logic 5hrs

<table>
<tr><th></th><th>Assessment statement</th><th>Teaching note</th><th>Obj</th></tr>
<tr><td>4.2.1</td><td>Define the Boolean operators and, or, not, nand, nor and xor, by drawing the appropriate truth table.</td><td></td><td>1</td></tr>
<tr><td>4.2.2</td><td>Construct Boolean expressions using the operators in 4.2.1.</td><td><table><tr><th>Operator</th><th>Symbol</th></tr><tr><td>and</td><td>•</td></tr><tr><td>or</td><td>+</td></tr><tr><td>not</td><td>(overbar)</td></tr><tr><td>xor</td><td>⊕</td></tr></table>
For example, $\left(A \oplus \overline{B}\right) \cdot \left(\overline{C + D}\right)$.

This can be written in words as: (A xor not B) and (C nor D).</td><td>4</td></tr>
<tr><td>4.2.3</td><td>Calculate the values of a Boolean expression using truth tables.</td><td>A maximum of three inputs will be expected. Include the use of truth tables to determine whether two Boolean expressions are logically equivalent.</td><td>3</td></tr>
<tr><td>4.2.4</td><td>Convert Boolean expressions into simpler forms.</td><td>A maximum of three inputs will be expected. Conversions may be done “algebraically” (using identities such as $x + 1 = 1$ and De Morgan’s laws) or by using Karnaugh maps, Venn diagrams or any other appropriate method.</td><td>2</td></tr>
<tr><td>4.2.5</td><td>Construct a simple logic circuit that corresponds to a given Boolean expression by using standard logic gates.</td><td></td><td>4</td></tr>
</table>

© International Baccalaureate Organization 2004

Topic 4—Computer mathematics and logic (continued)

	Assessment statement	Teaching note	Obj
4.2.6	Construct a Boolean expression that corresponds to a given logic circuit.		**4**
4.2.7	Explain the function of a given circuit.		**3**

© International Baccalaureate Organization 2004

Topic 5—Abstract data structures and algorithms

The Java programming language provides some standard data structures (such as arrays or files) that are adequate for many standard problems. Other problems require further data types to represent more complex structures, improve algorithm efficiency, or provide for more sophisticated memory management.

Although Java implements many different types of container class for the convenience of programmers, students are expected to be able to develop their own ADTs from first principles.

Higher level students must demonstrate mastery of some of these techniques in the program dossier and should be able to use any of these techniques during the examination. This topic extends several aspects of topics 1 and 2.

5.1—Fundamentals 3 hrs

	Assessment statement	Teaching note	Obj
5.1.1	Define operator (unary and binary), identifier, operand, actual parameter (argument), formal parameter, infix notation, postfix notation and prefix notation.		**1**
5.1.2	Define stack, queue and binary tree.		**1**
5.1.3	Discuss the features and appropriate usage of stacks including: parameter storage, interrupt handling, evaluation of arithmetic expressions and storage of subprogram return addresses.		**3**
5.1.4	Discuss the features and appropriate usage of queues including: keyboard queues, print queues and customer queue simulations.		**3**
5.1.5	Discuss the features and appropriate usage of binary trees including: storing search keys, decision trees and file systems.		**3**

© International Baccalaureate Organization 2004

Topic 5—Abstract data structures and algorithms (continued)

5.2—Static data structures 8 hrs

Arrays are covered at length in the common core. This subtopic should be considered as an extension.

	Assessment statement	Teaching note	Obj
5.2.1	Trace algorithms that perform a quicksort on linear arrays.		**2**
5.2.2	Construct algorithms that perform a quicksort on linear arrays.		**4**
5.2.3	Construct a hash table including the generation of addresses using modulo arithmetic and the handling of clashes by locating next free space.	Students will be given a hash algorithm and a set of keys or records that will be allocated memory locations by employing the algorithm.	**4**
5.2.4	Trace algorithms that implement a stack in an array.		**2**
5.2.5	Construct algorithms that implement a stack in an array.	This includes: to initialize a stack, to test for an empty or a full stack, to push a data item, to pop a data item and to display the top data item. All operations must protect against overflow and underflow.	**4**
5.2.6	Trace algorithms that implement a queue in an array.		**2**
5.2.7	Construct algorithms that implement a queue in an array.	This includes: to initialize a queue, to test for an empty or a full queue, to add a data item to the rear of a queue (enqueue), to remove a data item from the front of a queue (dequeue) and to display the data item at the front of a queue. Algorithms must include linear and circular implementation. All operations must protect against overflow and underflow.	**4**

© International Baccalaureate Organization 2004

Topic 5—Abstract data structures and algorithms (continued)

5.3—Dynamic data structures 14 hrs

	Assessment statement	Teaching note	Obj
5.3.1	Define object reference.		**1**
5.3.2	Construct algorithms that use reference mechanisms.		**4**
5.3.3	Discuss the features and appropriate usage of single, double and circular linked lists.		**3**
5.3.4	Outline and illustrate how links operate logically.		**2**
5.3.5	Trace algorithms to implement linked lists.		**2**
5.3.6	Construct algorithms to implement linked lists.	This includes: initialize, add objects, delete objects, find tail object, perform linear search and insert objects into a list. All operations must protect against null pointer exceptions.	**4**
5.3.7	Trace algorithms that implement a dynamic stack using references.		**2**
5.3.8	Construct algorithms that implement a dynamic stack using references.	Students must recognize the difference between this and the static representation of stacks. See notes in 5.2.5.	**4**
5.3.9	Trace algorithms that implement a dynamic queue using references.		**2**
5.3.10	Construct algorithms that implement a dynamic queue using references.	Students must recognize the difference between this and the static representation of a queue. See also notes in 5.2.7.	**4**
5.3.11	Define parent, left-child, right-child and subtree.		**1**
5.3.12	Trace algorithms to implement binary trees.		**2**

© International Baccalaureate Organization 2004

Topic 5—Abstract data structures and algorithms (continued)

	Assessment statement	Teaching note	Obj
5.3.13	Construct algorithms to implement binary trees.	This includes: initialize, add objects, traverse (pre-order, in-order and post-order). All tree traversals must be implemented recursively. See 5.5.	**4**
5.3.14	Outline and illustrate the logical representation of dynamic data structures.		**2**

5.4—Objects in problem solutions 6 hrs

The scope of the topic is limited to features exemplified in Java. (See appendix 2.)

	Assessment statement	Teaching note	Obj
5.4.1	Outline the features of an object.	This should be limited to the following definition. An object is a combination of data and the operations that can be performed in association with that data. Each data part of an object is referred to as a data member while the operations can be referred to as methods. The current state of an object is stored in its data members and that state should only be changed or accessed through the methods. Common categories of operations include: the construction of objects; operations that either set (mutator methods) or return (accessor methods) the data members; operations unique to the data type; and operations used internally by the object.	**2**
5.4.2	Explain the basic features and advantages of encapsulation.	Encapsulation is the combination of data and the operations that act on the data into a single "program unit" called an object. The advantages are that it allows for information and data hiding.	**3**
5.4.3	Explain the basic features and advantages of information and data hiding.	Once encapsulated into an object both the data members and the details of the implementation of the member functions can be hidden. This allows the object to be used at an abstract level.	**3**

© International Baccalaureate Organization 2004

Topic 5—Abstract data structures and algorithms (continued)

	Assessment statement	Teaching note	Obj
5.4.4	Explain the basic features and advantages of polymorphism.	Polymorphism describes the situation in which the same operation can be applied to different objects, with each object behaving appropriately. The concepts of templates, virtual member functions and operator overloading are not required. Polymorphism allows objects to be used intuitively and it simplifies coding by making it generic.	**3**
5.4.5	Explain the basic features and advantages of inheritance.	Inheritance allows one object to be derived from another. The derived object has all the data members and member functions of the original object and any additional data member or member functions that are defined within it. Even previously defined functionality may be redefined with the appropriate functionality applied to the particular object that invokes it. In Java, all classes are subclasses of the object class. When functions (including constructors) are redefined in a derived object, they completely override the original function. Inheritance in Java is limited to one object being derived from another (one level of inheritance). Multiple inheritance is not supported by the Java language.	**3**
5.4.6	Trace an algorithm that includes objects.	This will include recording the behaviour and state of the objects.	**2**

5.5—Recursion 6 hrs

	Assessment statement	Teaching note	Obj
5.5.1	Define recursion.		**1**
5.5.2	Discuss the advantages and disadvantages of recursion.	Students must understand that for some applications a recursive procedure is short and elegant, and that a recursive solution is ideally suited for some algorithms. However, recursion is not suitable for most algorithms as non-recursive ones are more efficient.	**3**

© International Baccalaureate Organization 2004

Topic 5—Abstract data structures and algorithms (continued)

	Assessment statement	Teaching note	Obj
5.5.3	Trace recursive algorithms.	All steps and calls must be shown clearly. Students may need to draw a tree.	**2**
5.5.4	Construct recursive algorithms.	This is limited to an algorithm that returns no more than one result and contains either one or two recursive calls to itself.	**4**
5.5.5	Implement the following constructs: self-referential classes and recursion.		**3**

5.6—Algorithm evaluation 4 hrs

	Assessment statement	Teaching note	Obj
5.6.1	State the efficiency of the following algorithms in BigO notation: a linear search is O(n), a bubble sort is $O(n^2)$, a quicksort is O(n log n), a binary search is O(log n) and a selection sort is $O(n^2)$, given a randomly distributed data set.	BigO notation is used to classify algorithm performance (speed). A sequential search is O(n), meaning that the time to search an array is proportional to the size of the array. However, a bubble sort requires nested loops and is therefore $O(n^2)$, so its time requirements are proportional to the square of the size of the list. Students should be aware that the efficiency of a given algorithm may depend on the distribution of the data, for example, a quicksort may deteriorate to $O(n^2)$ in the worst case.	**1**
5.6.2	Analyse the efficiency of algorithms (those in 5.6.1 and those of similar complexity), in terms of BigO notation and in terms of the storage requirements.	When students are presented with an algorithm that they have not encountered they must be able to write the BigO notation for the efficiency of that algorithm.	**3**

© International Baccalaureate Organization 2004

Topic 5—Abstract data structures and algorithms (continued)

	Assessment statement	Teaching note	Obj
5.6.3	Outline how data structures in this syllabus can be organized to suit the requirements of applications.	Students should consider the needs of different applications for data types and data structures. For example, **stacks** might be used to track changes to a word processing document whereas a **queue** might store data items being entered at the keyboard for subsequent processing in the order in which they arrived. Ordered **binary trees** and **hash tables** are frequently used to store key fields used for quick retrieval of items from an unordered **data file**.	**2**
5.6.4	Evaluate algorithms that use any of the data structures in this syllabus.	The algorithms may be standard algorithms mentioned in the syllabus, or algorithms of equivalent complexity that the students have not seen before.	**3**

© International Baccalaureate Organization 2004

Topic 6—Further system fundamentals

Actual computer system performance is affected by all the components of the system. Students need to know the functions of the individual components and the methods used in their interactions. This topic extends topic 3.

6.1—Processor configuration 2 hrs

	Assessment statement	Teaching note	Obj
6.1.1	Describe the functions of the following processor components: accumulator, instruction register and program counter.	Further details (or registers) are not required.	**2**
6.1.2	Explain the role of the above components in the execution of single instructions in the machine instruction cycle.		**3**
6.1.3	Describe the function of an interrupt register.		**2**
6.1.4	Describe how buses link the processor, the random access memory, the read-only memory and cache.		**2**

6.2—Magnetic disk storage 1 hr

	Assessment statement	Teaching note	Obj
6.2.1	Outline storage details with reference to blocking, sectors, cylinders and heads.		**2**
6.2.2	Describe access time in terms of latency (rotational delay), seek time and transfer time.		**2**

© International Baccalaureate Organization 2004

Topic 6—Further system fundamentals (continued)

6.3—Operating systems and utilities 2 hrs

	Assessment statement	Teaching note	Obj
6.3.1	Define operating system.	Knowledge of any specific operating system is not required.	**1**
6.3.2	Explain the functions of operating systems.	Students need to be aware that an operating system is a collection of programs that cover the following tasks: input/output (I/O) control, file maintenance, software/hardware interface, memory management, user interface, software execution control, security. Virtual memory must be included but knowledge of thrashing and paging is not required. This extends 3.3.2.	**3**
6.3.3	Outline the functions of linker, loader and library manager.		**2**

6.4—Further network fundamentals 4 hrs

	Assessment statement	Teaching note	Obj
6.4.1	Outline the role of the computers used in the separate type of networks: WAN, LAN and the Internet.	The roles of providers, servers and clients should be understood for each of these networks. Students should be able to select the appropriate type of network for a given situation. They must understand the role of gateways.	**2**
6.4.2	Describe the features of communications needed for networking.	Ethernet, public and private telephone lines, ISDN, ADSL, fibre optic and wireless methods should all be familiar and students should be able to select the most suitable method of communication in a given situation, and to state the advantages of each method. Technical details will not be required.	**2**
6.4.3	Describe packet switching.	Students need to be aware that when a message is dissembled into packets, the packets may take different paths and pass through different nodes to arrive at the same destination, and that packets can be discarded. Virtual circuits are not required.	**2**

© International Baccalaureate Organization 2004

Topic 6—Further system fundamentals (continued)

	Assessment statement	Teaching note	Obj
6.4.4	Outline the need for protocols in packet switching.	Students do not need to know technical details of TCP, IP, OSI, but must understand that protocols include essential information that allows packets to be reassembled at their destination according to the requirements of the receiving computer.	**2**
6.4.5	Explain the need for network security and describe how this can be achieved.	Emphasize the importance of protection within a LAN by giving layered access (for example, via permissions on certain areas) to different users, and marking files as read only. The need for a firewall to prevent intrusion from outside should be clear.	**3**

6.5—Computer/peripheral communication 6 hrs

	Assessment statement	Teaching note	Obj
6.5.1	Define port and handshaking.		**1**
6.5.2	Define direct memory access (DMA) and buffer.		**1**
6.5.3	Define interrupt and polling.		**1**
6.5.4	Explain how peripheral devices are controlled with reference to the printer, modem and disk drive.	This must include the use of buffers (including double buffering), interrupts and interrupt priorities, polling, direct memory access (DMA) and handshaking in these devices.	**3**
6.5.5	Compare the features of DMA, interrupt systems and polling systems.	Students must cover an event or external device interrupt as well as a polling system. Knowledge of specific interrupt codes is not required.	**3**
6.5.6	Compare serial transmission with parallel transmission.		**3**

© International Baccalaureate Organization 2004

Topic 7—File organization

A variety of file structures are commonly used in computer systems. Students must be familiar with several of the most common structures. This topic extends topic 1.

Current literature often appears confusing as the terminology relating to file structure and methods of accessing files is used inconsistently. The following table clarifies the specific terminology that is used in this syllabus, and that will be used in examinations.

File structure name	Structure details	Access method (searching)
Sequential file	Ordered or unordered records	Sequential access.
Partially-indexed file	Ordered records	Sequential access to index, followed by direct access to the first record in the group, then sequential access to find the desired record.
Fully-indexed file	Unordered records	Sequential access to the index, followed by direct access to the data file.
Direct access file	Unordered or ordered records	A calculation provides the address (location) of a record, followed by direct access to the record.

7.1—File organization 10 hrs

	Assessment statement	Teaching note	Obj
7.1.1	Define the term "key field".		**1**
7.1.2	Outline sequential file organization on unordered records and how records can be retrieved by using sequential access via the key field.		**2**
7.1.3	Outline sequential file organization on ordered records and how records can be retrieved by using sequential access via the key field.		**2**
7.1.4	Outline partially-indexed sequential file organization.	A partially-indexed sequential file has ordered records, with a separate but partial index. Students should be able to describe how records can be retrieved via access to the index, followed by direct access to the first record in a group, followed by sequential access to locate the desired record.	**2**

© International Baccalaureate Organization 2004

Topic 7—File organization

	Assessment statement	Teaching note	Obj
7.1.5	Outline fully-indexed file organization.	A fully-indexed file has unordered records, with a separate and complete index. Students should be able to describe how records can be retrieved via access to the index, followed by direct access in the data file. Recall of multi level indexes is not required.	**2**
7.1.6	Outline direct access file organization.	A direct access file can have unordered records. Students should be able to outline how records can be retrieved via a calculation followed by direct access.	**2**
7.1.7	Outline the need for fixed- and variable-length fields and records, and how they are related to direct and sequential access methods.		**2**
7.1.8	Describe the use of hash algorithms to save and retrieve records in a direct access file.	Students should understand the use of modulo operators (mod, div) in the construction of a hash function. See also 5.2.3.	**2**
7.1.9	Compare the speed of access and storage requirements for the types of files mentioned in 7.1.2–7.1.8.	This should also include storage media (disk, tape). Access speeds should be expressed in descriptions, calculations of iterations and BigO notation.	**3**
7.1.10	Explain how the logical organization of data differs from its physical organization.	For example, in a fully-indexed sequential file, records can be retrieved in alphabetical order by using the index, even though they are not stored physically in that order.	**3**
7.1.11	Outline the need for external sorts.	The sorting of files that are too large for the primary memory of a computer requires techniques based on a combination of sorting and merging. Recall of algorithms for merge sorts is not required.	**2**
7.1.12	Demonstrate an understanding of and use the different types of data streams identified in appendix 2.		**3**

© International Baccalaureate Organization 2004