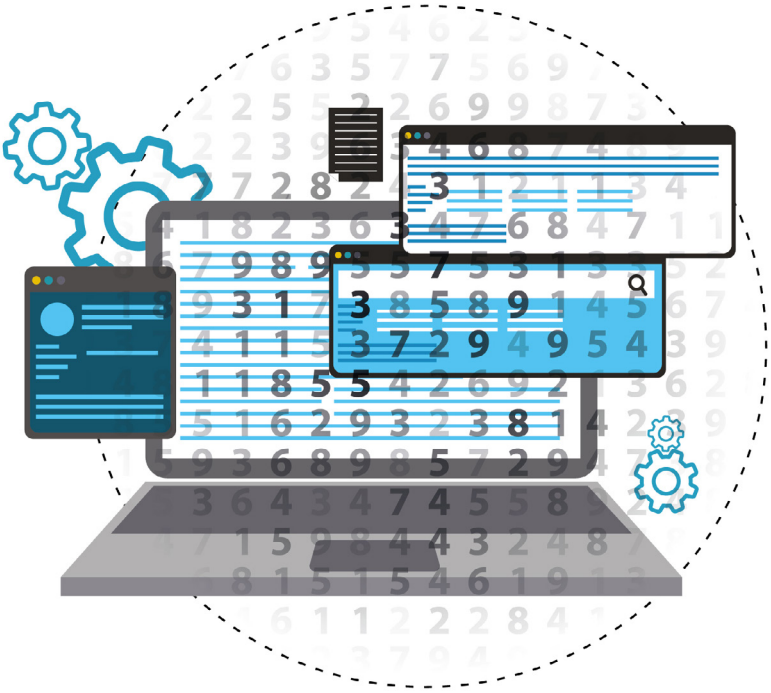
Cambridge Assessment International Education

Syllabus

Cambridge International AS & A Level Computer Science 9618

For examination in June and November 2021, 2022 and 2023.



Version 1

Cambridge Pathway

3 Subject content

AS Content

Computational thinking is developed using a structured approach that includes the use of programming and problem solving skills to provide solutions to real life problems. It requires the manipulation and storage of different types of data and the communication of solutions over networks.

Computational thinking is supported by developing an understanding of how computer architecture, hardware, systems software, security measures and communication systems, provide the infrastructure required in an efficient and ethical way. The syllabus supports opportunities for students to apply their skills in practical contexts that are required in the digital industry.

1 Information representation

1.1 Data Representation

Candidates should be able to:	Notes and guidance
Show understanding of binary magnitudes and the difference between binary prefixes and decimal prefixes	Understand the difference between and use: • kibi and kilo • mebi and mega • gibi and giga • tebi and tera
Show understanding of the basis of different number systems	Use the binary, denary, hexadecimal number bases and Binary Coded Decimal (BCD) and one's and two's complement representation for binary numbers Convert an integer value from one number base / representation to another
Perform binary addition and subtraction	Using positive and negative binary integers Show understanding of how overflow can occur
Describe practical applications where Binary Coded Decimal (BCD) and Hexadecimal are used	
Show understanding of and be able to represent character data in its internal binary form, depending on the character set used	Familiar with ASCII (American Standard Code for Information Interchange), extended ASCII and Unicode. Students will not be expected to memorise any particular character codes

1.2 Multimedia

Graphics

Candidates should be able to:	Notes and guidance
Show understanding of how data for a bitmapped image are encoded	Use and understand the terms: *pixel*, *file header*, *image resolution*, *screen resolution*, *colour depth*, *bit depth*
Perform calculations to estimate the file size for a bitmap image	
Show understanding of the effects of changing elements of a bitmap image on the image quality and file size	Use the terms: *image resolution*, *colour depth*
Show understanding of how data for a vector graphic are encoded	Use the terms: *drawing object*, *property*, *drawing list*
Justify the use of a bitmap image or a vector graphic for a given task	

Sound

Candidates should be able to:	Notes and guidance
Show understanding of how sound is represented and encoded	Use the terms: *sampling*, *sampling rate*, *sampling resolution*, *analogue and digital data*
Show understanding of the impact of changing the sampling rate and resolution	Impact on file size and accuracy

1.3 Compression

Candidates should be able to:	Notes and guidance
Show understanding of the need for and examples of the use of compression	
Show understanding of lossy and lossless compression and justify the use of a method in a given situation	
Show understanding of how a text file, bitmap image, vector graphic and sound file can be compressed	Including the use of run-length encoding (RLE)

2 Communication

2.1 Networks including the internet

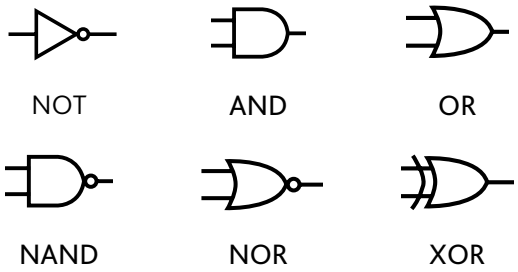
Candidates should be able to:	Notes and guidance
Show understanding of the purpose and benefits of networking devices	
Show understanding of the characteristics of a LAN (local area network) and a WAN (wide area network)	
Explain the client-server and peer-to-peer models of networked computers	Roles of the different computers within the network and subnetwork models Benefits and drawbacks of each model Justify the use of a model for a given situation
Show understanding of thin-client and thick-client and the differences between them	
Show understanding of the bus, star, mesh and hybrid topologies	Understand how packets are transmitted between two hosts for a given topology Justify the use of a topology for a given situation
Show understanding of cloud computing	Including the use of public and private clouds. Benefits and drawbacks of cloud computing
Show understanding of the differences between and implications of the use of wireless and wired networks	Describe the characteristics of copper cable, fibre-optic cable, radio waves (including WiFi), microwaves, satellites
Describe the hardware that is used to support a LAN	Including switch, server, Network Interface Card (NIC), Wireless Network Interface Card (WNIC), Wireless Access Points (WAP), cables, bridge, repeater
Describe the role and function of a router in a network	
Show understanding of Ethernet and how collisions are detected and avoided	Including Carrier Sense Multiple Access / Collision Detection (CSMA / CD)
Show understanding of bit streaming	Methods of bit streaming, i.e. real-time and on-demand Importance of bit rates / broadband speed on bit streaming
Show understanding of the differences between the World Wide Web (WWW) and the internet	
Describe the hardware that is used to support the internet	Including modems, PSTN (Public Switched Telephone Network), dedicated lines, cell phone network

2.1 Networks including the internet continued	
Explain the use of IP addresses in the transmission of data over the internet	Including: • format of an IP address including IPv4 and IPv6 • use of subnetting in a network • how an IP address is associated with a device on a network • difference between a public IP address and a private IP address and the implications for security • difference between a static IP address and a dynamic IP address
Explain how a Uniform Resource Locator (URL) is used to locate a resource on the World Wide Web (WWW) and the role of the Domain Name Service (DNS)	

3 Hardware

3.1 Computers and their components	
Candidates should be able to:	Notes and guidance
Show understanding of the need for input, output, primary memory and secondary (including removable) storage	
Show understanding of embedded systems	Including: benefits and drawbacks of embedded systems
Describe the principal operations of hardware devices	Including: Laser printer, 3D printer, microphone, speakers, magnetic hard disk, solid state (flash) memory, optical disc reader/writer, touchscreen, virtual headset
Show understanding of the use of buffers	
Explain the differences between Random Access Memory (RAM) and Read Only Memory (ROM)	Including their use in a range of devices and systems
Explain the differences between Static RAM (SRAM) and Dynamic RAM (DRAM)	Include their use in a range of devices and systems and the reasons for using one instead of the other depending on the device and its use
Explain the difference between Programmable ROM (PROM), Erasable Programmable ROM (EPROM) and Electrically Erasable Programmable ROM (EEPROM)	
Show an understanding of monitoring and control systems	Including: • difference between monitoring and control • use of sensors (including temperature, pressure, infra-red, sound) and actuators • importance of feedback

3.2 Logic Gates and Logic Circuits

Candidates should be able to:	Notes and guidance
Use the following logic gate symbols:  	
Understand and define the functions of : NOT, AND, OR, NAND, NOR and XOR (EOR) gates	All gates except the NOT gate will have two inputs only.
Construct the truth table for each of the logic gates above	
Construct a logic circuit	From: • a problem statement • a logic expression • a truth table
Construct a truth table	From: • a problem statement • a logic circuit • a logic expression
Construct a logic expression	From: • a problem statement • a logic circuit • a truth table

4 Processor Fundamentals

4.1 Central Processing Unit (CPU) Architecture

Candidates should be able to:	Notes and guidance
Show understanding of the basic Von Neumann model for a computer system and the stored program concept	
Show understanding of the purpose and role of registers, including the difference between general purpose and special purpose registers	Special purpose registers including: • Program Counter (PC) • Memory Data Register (MDR) • Memory Address Register (MAR) • The Accumulator (ACC) • Index Register (IX) • Current Instruction Register (CIR) • Status Register
Show understanding of the purpose and roles of the Arithmetic and Logic Unit (ALU), Control Unit (CU) and system clock, Immediate Access Store (IAS)	
Show understanding of how data are transferred between various components of the computer system using the address bus, data bus and control bus	
Show understanding of how factors contribute to the performance of the computer system	Including: • processor type and number of cores • the bus width • clock speed • cache memory
Understand how different ports provide connection to peripheral devices	Including connection to: • Universal Serial Bus (USB) • High Definition Multimedia Interface (HDMI) • Video Graphics Array (VGA)
Describe the stages of the Fetch-Execute (F-E) cycle	Describe and use 'register transfer' notation to describe the F–E cycle
Show understanding of the purpose of interrupts	Including: • possible causes of interrupts • applications of interrupts • use of an Interrupt service (ISR) handling routine • when interrupts are detected during the fetch-execute cycle • how interrupts are handled

4.2 Assembly Language

Candidates should be able to:	Notes and guidance
Show understanding of the relationship between assembly language and machine code	
Describe the different stages of the assembly process for a two-pass assembler	Apply the two-pass assembler process to a given simple assembly language program
Trace a given simple assembly language program	
Show understanding that a set of instructions are grouped	Including the following groups: • Data movement • Input and output of data • Arithmetic operations • Unconditional and conditional instructions • Compare instructions
Modes of addressing	Including Immediate, direct, indirect, indexed, relative

The following table is an example of an instruction set:

Instruction		Explanation
Opcode	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC
LDR	#n	Immediate addressing. Load the number n to IX
MOV	<register>	Move the contents of the accumulator to the given register (IX)
STO	<address>	Store the contents of ACC at the given address
ADD	<address>	Add the contents of the given address to the ACC
ADD	#n	Add the denary number n to the ACC
SUB	<address>	Subtract the contents of the given address from the ACC
SUB	#n	Subtract the denary number n from the ACC
INC	<register>	Add 1 to the contents of the register (ACC or IX)
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX)
JMP	<address>	Jump to the given address
CMP	<address>	Compare the contents of ACC with the contents of <address>
CMP	#n	Compare the contents of ACC with number n
CMI	<address>	Indirect addressing. The address to be used is at the given address. Compare the contents of ACC with the contents of this second address
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False
IN		Key in a character and store its ASCII value in ACC
OUT		Output to the screen the character whose ASCII value is stored in ACC
END		Return control to the operating system

All questions will assume there is only one general purpose register available (Accumulator)

ACC denotes Accumulator

IX denotes Index Register

# denotes immediate addressing
B denotes a binary number, e.g. B01001010
& denotes a hexadecimal number, e.g. &4A

4.3 Bit manipulation

Candidates should be able to:	Notes and guidance
Show understanding of and perform binary shifts	logical, arithmetic and cyclic Left shift, right shift
Show understanding of how bit manipulation can be used to monitor / control a device	Carry out bit manipulation operations Test and set a bit (using bit masking)

Label	Instruction		Explanation
	Opcode	Operand	
	AND	#n	Bitwise AND operation of the contents of ACC with the operand
	AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>
	XOR	#n	Bitwise XOR operation of the contents of ACC with the operand
	XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>
	OR	#n	Bitwise OR operation of the contents of ACC with the operand
	OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address> <address> can be an absolute address or a symbolic address
	LSL	#n	Bits in ACC are shifted logically n places to the left. Zeros are introduced on the right hand end
	LSR	#n	Bits in ACC are shifted logically n places to the right. Zeros are introduced on the left hand end
<label>:	<opcode>	<operand>	Labels an instruction
<label>:		<data>	Gives a symbolic address <label> to the memory location with contents <data>

5 System Software

5.1 Operating System

Candidates should be able to:	Notes and guidance
Explain why a computer system requires an Operating System (OS)	
Explain the key management tasks carried out by the Operating System	Including memory management, file management, security management, hardware management (input / output / peripherals), process management
Show understanding of the need for typical utility software provided with an Operating System	Including disk formatter, virus checker, defragmentation software, disk contents analysis/disk repair software, file compression, back-up software
Show understanding of program libraries	Including: • software under development is often constructed using existing code from program libraries • the benefits to the developer of software constructed using library files, including Dynamic Link Library (DLL) files

5.2 Language Translators

Candidates should be able to:	Notes and guidance
Show understanding of the need for: • assembler software for the translation of an assembly language program • a compiler for the translation of a high-level language program • an interpreter for translation and execution of a high-level language program	
Explain the benefits and drawbacks of using either a compiler or interpreter and justify the use of each	
Show awareness that high-level language programs may be partially compiled and partially interpreted, such as Java	
Describe features found in a typical Integrated Development Environment (IDE)	Including: • for coding, including context-sensitive prompts • for initial error detection, including dynamic syntax checks • for presentation, including prettyprint, expand and collapse code blocks • for debugging, including single stepping, breakpoints, variable expression, report window

6 Security, privacy and data integrity

6.1 Data Security

Candidates should be able to:	Notes and guidance
Explain the difference between the terms security, privacy and integrity of data	
Show appreciation of the need for both the security of data and the security of the computer system	
Describe security measures designed to protect computer systems, ranging from the stand-alone PC to a network of computers	Including user accounts, passwords, authentication techniques such as digital signatures, firewall, anti-virus software, anti-spyware, encryption
Show understanding of the threats to computer and data security posed by networks and the internet	Including malware (virus, spyware), hackers, phishing, pharming
Describe methods that can be used to restrict the risks posed by threats	
Describe security methods designed to protect the security of data	Including encryption, access rights

6.2 Data Integrity

Candidates should be able to:	Notes and guidance
Describe how data validation and data verification help protect the integrity of data	
Describe and use methods of data validation	Including range check, format check, length check, presence check, existence check, limit check, check digit
Describe and use methods of data verification during data entry and data transfer	During data entry including visual check, double entry During data transfer including parity check (byte and block), checksum

7 Ethics and Ownership

7.1 Ethics and Ownership

Candidates should be able to:	Notes and guidance
Show understanding of the need for and purpose of ethics as a computing professional	Understand the importance of joining a professional ethical body including BCS (British Computer Society), IEEE (Institute of Electrical and Electronic Engineers)
Show understanding of the need to act ethically and the impact of acting ethically or unethically for a given situation	
Show understanding of the need for copyright legislation	
Show understanding of the different types of software licencing and justify the use of a licence for a given situation	Licences to include free Software Foundation, the Open Source Initiative, shareware and commercial software
Show understanding of Artificial Intelligence (AI)	Understand the impact of AI including social, economic and environmental issues Understand the applications of AI

8 Databases

8.1 Database Concepts

Candidates should be able to:	Notes and guidance
Show understanding of the limitations of using a file-based approach for the storage and retrieval of data	
Describe the features of a relational database that address the limitations of a file-based approach	
Show understanding of and use the terminology associated with a relational database model	Including entity, table, record, field, tuple, attribute, primary key, candidate key, secondary key, foreign key, relationship (one-to-many, one-to-one, many-to-many), referential integrity, indexing
Use an entity-relationship (E-R) diagram to document a database design	
Show understanding of the normalisation process	First Normal Form(1NF), Second Normal Form (2NF) and Third Normal Form (3NF)
Explain why a given set of database tables are, or are not, in 3NF	
Produce a normalised database design for a description of a database, a given set of data, or a given set of tables	

8.2 Database Management System (DBMS)

Candidates should be able to:	Notes and guidance
Show understanding of the features provided by a Database Management System (DBMS) that address the issues of a file based approach	Including: • data management, including maintaining a data dictionary • data modelling • logical schema • data integrity • data security, including backup procedures and the use of access rights to individuals / groups of users
Show understanding of how software tools found within a DBMS are used in practice	Including the use and purpose of: • developer interface • query processor

8.3 Data Definition Language (DDL) and Data Manipulation Language (DML)

Candidates should be able to:	Notes and guidance
Show understanding that DBMS carries out all creation / modification of the database structure using its Data Definition Language (DDL)	
Show understanding that the DBMS carries out all queries and maintenance of data using its DML	
Show understanding that the industry standard for both DDL and DML is Structured Query Language (SQL)	Understand a given SQL script
Understand given SQL (DDL) commands and be able to write simple SQL (DDL) commands using a sub-set of commands	Create a database (CREATE DATABASE) Create a table definition (CREATE TABLE), including the creation of attributes with appropriate data types: • CHARACTER • VARCHAR(n) • BOOLEAN • INTEGER • REAL • DATE • TIME change a table definition (ALTER TABLE) add a primary key to a table (PRIMARY KEY (field)) add a foreign key to a table (FOREIGN KEY (field) REFERENCES Table (Field))
Write an SQL script to query or modify data (DML) which are stored in (at most two) database tables	Queries including SELECT, FROM, WHERE, ORDER BY, GROUP BY, INNER JOIN, SUM, COUNT, AVG Data maintenance including. INSERT INTO, DELETE FROM, UPDATE

9 Algorithm Design and Problem Solving

Refer to Pseudocode Guide **www.cambridgeinternational.org/support**

9.1 Computational Thinking Skills

Candidates should be able to:	Notes and guidance
Show an understanding of abstraction	Need for and benefits of using abstraction Describe the purpose of abstraction, Produce an abstract model of a system by only including essential details
Describe and use decomposition	Break down problems into sub-problems leading to the concept of a program module (procedure / function)

9.2 Algorithms

Candidates should be able to:	Notes and guidance
Show understanding that an algorithm is a solution to a problem expressed as a sequence of defined steps	
Use suitable identifier names for the representation of data used by a problem and represent these using an identifier table	
Write pseudocode that contains input, process and output	
Write pseudocode using the four basic constructs of assignment, sequence, selection and repetition	
Document a simple algorithm using pseudocode	
Write pseudocode from: • a structured English description • a flowchart	
Describe and use the process of stepwise refinement to express an algorithm to a level of detail from which the task may be programmed	
Use logic statements to define parts of an algorithm solution	

10 Data Types and structures

10.1 Data Types and Records

Candidates should be able to:	Notes and guidance
Select and use appropriate data types for a problem solution	including integer, real, char, string, Boolean, date (pseudocode will use the following data types: INTEGER, REAL, CHAR, STRING, BOOLEAN, DATE, ARRAY, FILE)
Show understanding of the purpose of a record structure to hold a set of data of different data types under one identifier	Write pseudocode to define a record structure. Write pseudocode to read data from a record structure and save data to a record structure

10.2 Arrays

Candidates should be able to:	Notes and guidance
Use the technical terms associated with arrays	Including index, upper and lower bound
Select a suitable data structure (1D or 2D array) to use for a given task	
Write pseudocode for 1D and 2D arrays	
Write pseudocode to process array data	Sort using a bubble sort Search using a linear search

10.3 Files

Candidates should be able to:	Notes and guidance
Show understanding of why files are needed	
Write pseudocode to handle text files that consist of one or more lines	

10.4 Introduction to Abstract Data Types (ADT)

Candidates should be able to:	Notes and guidance
Show understanding that an ADT is a collection of data and a set of operations on those data	
Show understanding that a stack, queue and linked list are examples of ADTs	Describe the key features of a stack, queue and linked list and justify their use for a given situation
Use a stack, queue and linked list to store data	Candidates will not be required to write pseudocode for these structures, but they should be able to add, edit and delete data from these structures
Describe how a queue, stack and linked list can be implemented using arrays	

11 Programming

11.1 Programming Basics

Candidates should be able to:	Notes and guidance
Implement and write pseudocode from a given design presented as either a program flowchart or structured English	
Write pseudocode statements for: • the declaration of variables and constants • the assignment of values to variables and constants • expressions involving any of the arithmetic or logical operators input from the keyboard and output to the console	
Use built-in functions and library routines	Any functions not given in the pseudocode guide will be provided String manipulation functions will always be given

11.2 Constructs

Candidates should be able to:	Notes and guidance
Use pseudocode to write: • an 'IF' structure including the 'ELSE' clause and nested IF statements • a 'CASE' structure • a 'count-controlled' loop: • a 'post-condition' loop • a 'pre-condition' loop	
Justify why one loop structure may be better suited to solve a problem than the others	

11.3 Structured Programming

Candidates should be able to:	Notes and guidance
Define and use a procedure	
Explain where in the construction of an algorithm it would be appropriate to use a procedure	
Use parameters	A procedure may have none, one or more parameters A parameter can be passed by reference or by value
Define and use a function	
Explain where in the construction of an algorithm it is appropriate to use a function	A function is used in an expression, e.g. the return value replaces the call
Use the terminology associated with procedures and functions	including Procedure / function header, procedure / function interface, parameter, argument, return value
Write efficient pseudocode	

12 Software Development

12.1 Program Development Life cycle

Candidates should be able to:	Notes and guidance
Show understanding of the purpose of a development life cycle	
Show understanding of the need for different development life cycles depending on the program being developed	Including, waterfall, iterative, rapid application development (RAD)
Describe the principles, benefits and drawbacks of each type of life cycle	
Show understanding of the analysis, design, coding, testing and maintenance stages in the program development life cycle	

12.2 Program Design

Candidates should be able to:	Notes and guidance
Use a structure chart to decompose a problem into sub-tasks and express the parameters passed between the various modules / procedures / functions which are part of the algorithm design	Describe the purpose of a structure chart Construct a structure chart for a given problem Derive equivalent pseudocode from a structure chart
Show understanding of the purpose of state-transition diagrams to document an algorithm	

12.3 Program Testing and maintenance

Candidates should be able to:	Notes and guidance
Show understanding of ways of exposing and avoiding faults in programs	
Locate and identify the different types of errors	• syntax errors • logic errors • run-time errors
Correct identified errors	
Show understanding of the methods of testing available and select appropriate data for a given method	Including dry run, walkthrough, white-box, black-box, integration, alpha, beta, acceptance, stub
Show understanding of the need for a test strategy and test plan and their likely contents	
Choose appropriate test data for a test plan	Including normal, abnormal and extreme/boundary
Show understanding of the need for continuing maintenance of a system and the differences between each type of maintenance	Including perfective, adaptive, corrective
Analyse an existing program and make amendments to enhance functionality	