International Advanced Level
Computer Science

Specification

Pearson Edexcel International Advanced Subsidiary in Computer Science (XCP01)
Pearson Edexcel International Advanced Level in Computer Science (YCP01)

First teaching September 2026 | First examination June 2027

First certification from August 2027 (International Advanced Subsidiary)
and August 2028 (International Advanced Level)

Issue 1

Pearson

Contents

1 Introduction

Pearson Edexcel International Advanced Subsidiary in Computer Science and Pearson Edexcel International Advanced Level in Computer Science are part of a suite of International Advanced Level qualifications offered by Pearson.

These qualifications are not accredited or regulated by any UK regulatory body.

Key features

Pearson Edexcel International Advanced Subsidiary in Computer Science and the Pearson Edexcel International Advanced Level in Computer Science are modular qualifications. The Advanced Subsidiary can be obtained on completion of the International Advanced Subsidiary (IAS) units. The International Advanced Level can be obtained on completion of all the units (IAS and IA2).

Content

The content is relevant and engaging for learners who want to study this subject at a higher level. The qualification includes theoretical content that provides learners with a comprehensive coverage of fundamental topics and an industry-ready programming focus with hands-on experience of three programming paradigms using the programming language Python.

Assessment

The International Advanced Subsidiary consists of two external assessments, and the International Advanced Level consists of two further external assessments. In each qualification, one unit is a theory examination that is available either paper-based or onscreen, and the second unit requires learners to carry out practical programming using a computer.

Approach

Learners will develop their knowledge and understanding of computer science by applying the concepts outlined in this specification to a diverse set of problems and scenarios that reflect aspects of the real world. They will also enhance their practical skills, with a particular focus on programming and developing efficient computational solutions.

Specification updates

This specification is Issue 1 and is valid for first teaching from September 2026. If there are any significant changes to the specification, we will communicate these to centres through our regular communication channels, including notices on our website.

For more information, please visit qualifications.pearson.com.

Qualification aims and objectives

The aims and objectives of this qualification are to enable learners to:

- understand and apply key principles of computer science, including computational thinking, logic, algorithms, data representation and abstraction
- develop problem-solving skills by designing, writing, testing and refining programs using Python 3 within a suitable integrated development environment (IDE)
- apply programming techniques and system knowledge to develop practical and effective solutions
- build knowledge of computer systems, architecture, networks, cybersecurity and data handling in both theoretical and practical contexts
- think creatively and critically when exploring computing systems, networks, artificial intelligence (AI) and emerging technologies
- explore emerging technologies and understand the social, ethical, legal and environmental impact of computing in a global digital society
- communicate computing ideas effectively using appropriate technical terminology and structured reasoning
- gain transferable skills such as independent learning, critical thinking, and problem-solving, supporting progression to higher education and associated careers
- apply mathematical skills relevant to computer science.

Qualification abbreviations used in this specification

The following abbreviations appear in this specification:

- International Advanced Subsidiary — IAS
- International A2 — IA2 (the additional content required for an IAL)
- International Advanced Level — IAL.

Why choose Pearson Edexcel International Advanced Subsidiary/Advanced Level qualifications in Computer Science

A future-ready international qualification offering flexible onscreen or paper-based assessment, with a strong practical focus on programming.

We have listened to feedback from across the international school community, including teachers, higher education institutions and professionals in the field of Computer Science. We have used this opportunity to design our new International Advanced Subsidiary and Advanced Level qualifications in Computer Science to meet the evolving needs of learners around the world.

This qualification equips learners with both the theoretical understanding and the practical programming skills needed to thrive in today's fast-paced digital world. Through a combination of external examinations and innovative practical, onscreen assessments, learners will gain valuable hands-on experience and build the confidence to solve aspects of real-world problems using computational thinking.

This qualification builds on the core principles valued across our qualifications; these include:

Modular assessment structure

Supporting flexibility to sit examinations at different times of the year and providing opportunities to resit individual unit assessments before receiving an overall qualification grade. It is strongly recommended that learners follow a parallel approach to unit completion. For the IAS, learners should complete Unit 1 alongside Unit 2 during the first year of study. For IA2, learners should progress to Unit 3 alongside Unit 4 in the second year of study. Any resits should be taken after the completion of the relevant units.

Clear and simple structure

Combining written and practical elements that balance theory and practical applications, giving learners a rounded experience in computer science.

Range of question types

Designed to assess both the breadth of subject knowledge and the depth of learners understanding.

Progression

With a strong focus on skills that are relevant for university, employment, and the tech industry, our IAL Computer Science offers a future-ready qualification that prepares learners to succeed in the path they choose.

Key qualification features

Flexible theory assessment options

This innovative qualification supports the development of modern digital skills and offers assessments for Unit 1 and Unit 3 in either paper-based or onscreen formats. The assessment design promotes equitable access by balancing interactivity with inclusive practices. This approach ensures that learners with diverse needs are supported and provides centres with the flexibility to choose the assessment delivery method most appropriate for their learners.

Balanced and transparent assessment model

Our qualification has a straightforward structure, with each unit being fully externally assessed. All four units are equally weighted, and assessment durations are consistent across both IAS and IA2 levels. The three-hour practical programming exams reflect the time needed for realistic problem-solving, similar to professional practice in industry.

Practical programming experience

Our practical programming assessments for Unit 2 and Unit 4 are delivered fully onscreen to support programming in a live coding environment using an integrated development environment (IDE). It provides the learners with the opportunity to design, program and test programs. This practical approach helps gain hands-on coding experience and develop coding confidence.

Real world programming language

Our qualification develops learners' programming skills using Python 3, a widely used and industry-relevant language. By removing the reliance on pseudocode to capture or define algorithms, learners engage directly with authentic programming environments, supporting deeper understanding and development of transferable skills.

Programming Language Subset (PLS)

We provide a clear and accessible Programming Language Subset (PLS) document, outlining the primary constructs learners are expected to be familiar with and use, whilst not preventing them from using others. This offers structure and support while still allowing flexibility for learners to apply broader programming knowledge where appropriate.

Comprehensive teaching and learning support

A dedicated textbook supports the full course, providing clear, structured guidance for both teachers and learners. The content is closely aligned with the specification and covers both theoretical and practical elements. In addition, a range of teaching and learning materials are available to support effective preparation and delivery of the qualification.

Supporting you in planning and implementing this qualification

Planning

Our ***Getting Started Guide*** gives you an overview of the new International Advanced Subsidiary/Advanced Level in Computer Science qualifications to help you to get to grips with the content and assessment, and to help you understand what these mean for you and your learners.

Teaching and learning

There is a range of free teaching and learning support available to help you deliver the new qualifications, including:

- an editable course planner and schemes of work available to assist in developing teaching strategies for delivering the specification content. These resources can be tailored to meet the specific needs of your department
- additional support material, such as the Good Programming Practice Guide (GPPG) to help ensure that learners are developing the skills needed that form a key part of the assessment
- 'Getting Ready to Teach' and other training events, available on demand and live
- printed textbook and eBook resource to promote 'any time, any place' learning and offer additional support.

Find our resources on our website: qualifications.pearson.com.

Preparing for exams

We will also provide a range of resources to help you prepare your learners for the assessments, including:

- additional assessment materials to support formative assessments and mock exams
- marked exemplars of learner work with examiner commentaries
- examiner commentaries following each examination series
- additional support and training for the onscreen provision and practical programming examinations.

Onscreen assessment

Onscreen assessment is available for this qualification. Find out more information on our website: qualifications.pearson.com.

ResultsPlus

ResultsPlus provides the most detailed analysis available of your learners exam performance. It can help you identify the topics and skills where further learning would benefit your learners.

examWizard

This is an online data bank of past exam questions designed to support learners and teachers with exam preparation and assessment.

Training events

In addition to online training, we host a series of training events for teachers to deepen their understanding of our qualifications.

Get help and support

Our subject advisor ensures that you receive help and guidance from us. You can email our subject advisor at TeachingComputerScience@pearson.com. You can also sign up to receive subject advisor updates or contact us using our support portal.

Learn more at qualifications.pearson.com.

Qualification at a glance

Qualification overview

This modular qualification is designed to support progressive learning. It is strongly recommended that learners complete the unit assessments in the intended sequence for their initial assessment attempt. Subsequent resits may be taken in any order.

The International Advanced Level Unit 3 builds on knowledge and understanding developed in International Advanced Subsidiary Unit 1, extending learners' theoretical understanding of advanced computing principles and concepts.

The International Advanced Level Unit 4 requires learners to design and implement robust software solutions, applying the core principles and best practices introduced in International Advanced Subsidiary Unit 2.

Pearson Edexcel International **Advanced Subsidiary in Computer Science**

This qualification consists of two externally examined units.

The International Advanced Subsidiary (IAS) is the first half of the International Advanced Level qualification and consists of two IAS Units, 1 and 2. This qualification can be awarded as a discrete qualification or may contribute 50% towards the International Advanced Level qualification.

This qualification will include questions that target mathematics at level 2 or above. Overall, a minimum of 10% of the marks across the papers will be awarded for mathematics.

Pearson Edexcel International **Advanced Level in Computer Science**

This qualification consists of four externally examined units.

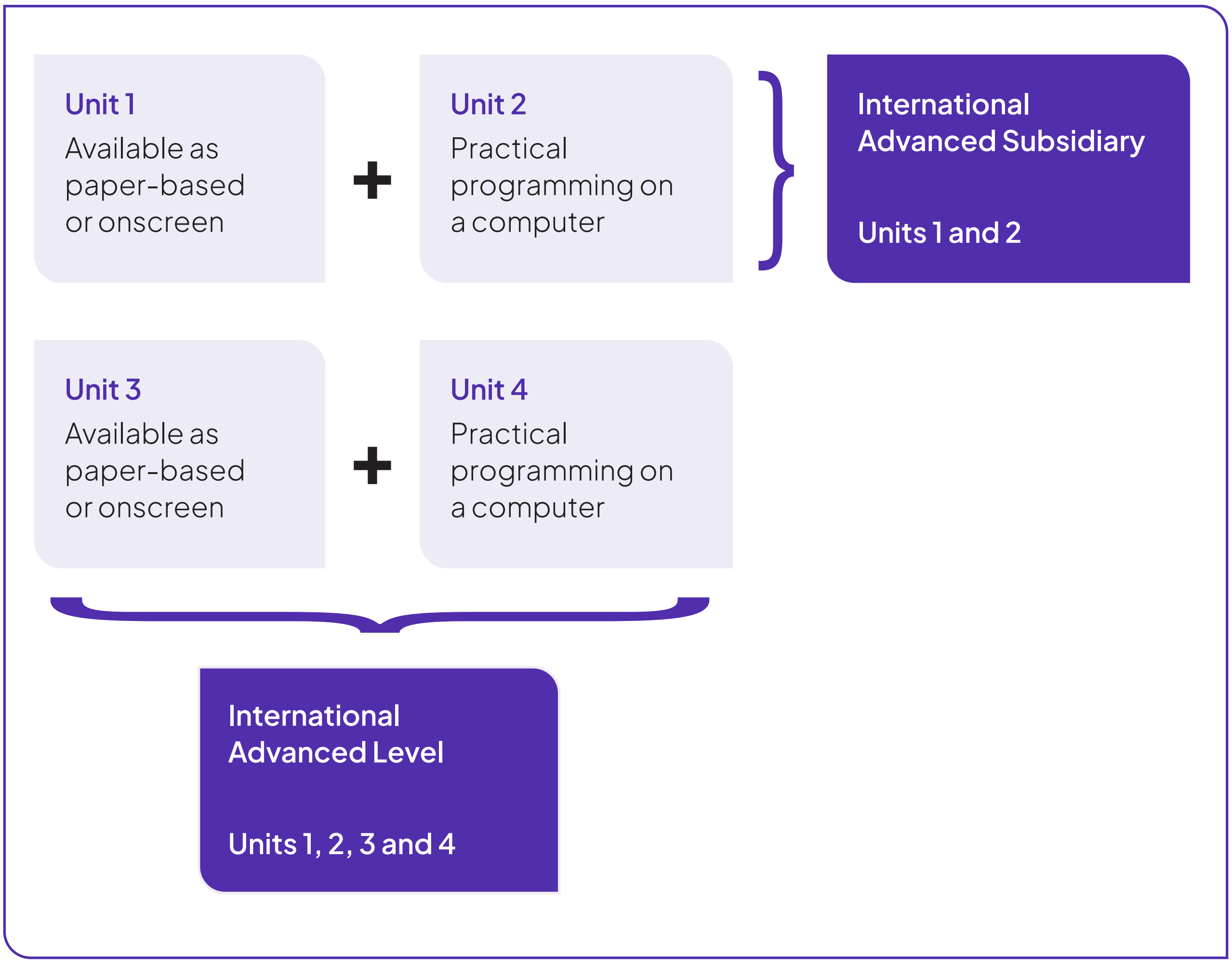
The International Advanced Level consists of the two IAS Units, 1 and 2, plus two IA2 Units, 3 and 4. Learners wishing to take the International Advanced Level must complete all four units.

This qualification will include questions that target mathematics at level 2 or above. Overall, a minimum of 10% of the marks across the papers will be awarded for mathematics.

Course introduction

Pearson Edexcel International Advanced Subsidiary and International Advanced Level in Computer Science is part of a suite of International Advanced Level qualifications offered by Pearson.

How the assessment relates to the qualifications available is shown below.


Unit 1
Available as paper-based or onscreen
Unit 2
Practical programming on a computer
International Advanced Subsidiary
Units 1 and 2
Unit 3
Available as paper-based or onscreen
Unit 4
Practical programming on a computer
International Advanced Level
Units 1, 2, 3 and 4

Course of study

The structure of these qualifications allows teachers to construct a course of study that can be taught and assessed as either:

- distinct modules of teaching and learning with related units of assessment taken at appropriate stages during the course; or
- a linear course assessed in its entirety at the end.

Content and assessment overview

Pearson Edexcel International **Advanced Subsidiary in Computer Science**.

Unit 1: Principles of Computer Science	Unit 2: Practical Programming and Problem-solving
Externally assessed examination: 1 hour and 30 minutes **Written (*Unit code: WCP01/01) or** **Onscreen (*Unit code: WCP01/01C)**	Externally assessed examination: 3 hours **Practical (*Unit code: WCP02/01)**
80 Marks 25% of the IAL 50% of the IAS	80 Marks 25% of the IAL 50% of the IAS
Available: January and June	**Available:** January and June
Content overview This unit assesses knowledge and understanding of the key principles of computer science. These will be assessed in topics 1 to 6. • Topic 1: Computer Systems • Topic 2: Data Representation • Topic 3: Networks and Encryption • Topic 4: Structuring Data • Topic 5: Problem-solving • Topic 6: Enabling Technologies	**Content overview** This unit assesses the practical application of computational thinking. These will be assessed in topics 7 to 10. • Topic 7: Programming • Topic 8: Organising and Handling Data • Topic 9: Best Practice • Topic 10: Computational Thinking and Algorithms
Assessment overview Learners must answer all questions. This assessment may include multiple choice, short/medium open response, extended open response, tabular and diagrammatic items. This assessment is available as paper-based or onscreen.	**Assessment overview** Learners must answer all questions. This practical assessment requires learners to design, write, test and refine a program or solve a problem using Python 3 in an integrated development environment (IDE) of choice. Learners will be provided with: • hard copy of the question paper • digital coding files • digital Programming Language Subset (PLS) document to reference during the assessment.

*See *Appendix 1: Codes* for a description of this code and all other codes relevant to this qualification.

Pearson Edexcel International **Advanced Level in Computer Science**.

Unit 3: Advanced Principles of Computer Science	**Unit 4: Advanced Practical Programming and Problem-solving**
Externally assessed examination: 1 hour and 30 minutes **Written (*Unit code: WCP03/01) or** **Onscreen (*Unit code: WCP03/01C)**	Externally assessed examination: 3 hours **Practical (*Unit code: WCP04/01)**
80 Marks 25% of the IAL 50% of the IA2	80 Marks 25% of the IAL 50% of the IA2
Available: January and June	**Available:** January and June
Content overview This unit assesses the practical applications of computational thinking. These will be assessed in topics 11 to 16. • Topic 11: Computer Systems and Data Representation • Topic 12: Networks and Cybersecurity • Topic 13: Programming Languages • Topic 14: Structuring Data • Topic 15: Problem-solving • Topic 16: Emerging Technologies and Professional Practice	**Content overview** This unit assesses practical applications of computational thinking. These will be assessed in topics 17 to 21. • Topic 17: Programming • Topic 18: Representing and Handling Data • Topic 19: Best Practice • Topic 20: Additional Programming Paradigms • Topic 21: Algorithms
Assessment overview Learners must answer all questions. This assessment may include multiple choice, short/medium open response, extended open response, tabular and diagrammatic items. The assembly language instruction set document will be available to learners to reference during the assessment. *See Appendix 3: Assembly language* instruction set. This assessment is available as paper-based or onscreen.	**Assessment overview** Learners must answer all questions. This practical assessment requires learners to design, write, test and refine a program or solve a problem using Python 3 in an integrated development environment (IDE) of choice. Learners will be provided with: • hard copy of the question paper • digital coding files • digital Programming Language Subset (PLS) document to reference during the assessment.

**See Appendix 1: Codes* for a description of this code and all other codes relevant to this qualification.

2 Subject content and assessment information

Learners are expected to demonstrate and apply the knowledge, understanding and skills described in the content. They are also expected to analyse, interpret and evaluate a range of information, using their knowledge, understanding and skills to solve programming problems.

Learners should develop their ability to apply appropriate computer-related mathematical skills throughout the course, for example, for data handling. This may involve, but is not limited to, working with units of measurements, using appropriate operators and using Boolean algebra. These skills will be integrated within topics and contexts of questions. Practical programming in the form of understanding coding is central to computer science.

Throughout the course, learners will develop and build on their practical skills.

Learners will be required to make decisions to design, write, test and refine a program or solve a problem, using Python 3 as the high-level programming language.

Learners will be assessed on these skills in Unit 2 and Unit 4, where they may be presented with familiar and unfamiliar scenarios.

Unit 1: Principles of Computer Science

Assessment information

First assessment: May/June 2027

The assessment is 1 hours and 30 minutes.

The assessment is out of 80 marks.

Learners must answer all questions.

This unit may include multiple choice, short/medium open response, extended open response, tabular and diagrammatic items.

Calculators must not be used in the examination.

Sample assessment materials

A sample paper and mark scheme for this unit can be found in the *Pearson Edexcel International Advanced Level in Computer Science Sample Assessment Materials (SAMs)* document.

A full list of command words that will be used in assessments is shown in *Appendix 5: Command word taxonomy.*

Overview

This unit develops learners' foundational understanding of essential computer science principles. Learners will explore how computers work, how data is represented, how networks operate, and how information is structured and secured. Emphasis is placed on computational thinking, logic, problem-solving strategies, and the enabling technologies that underpin digital systems. This unit prepares learners to analyse problems, understand systems and data, and make informed decisions about the application of computing technologies. Learners will be expected to apply their knowledge and understanding to familiar and unfamiliar contexts.

Topics 4 and 5 cover the programming theory underpinning the practical coding that is the focus of Unit 2.

Learners are expected to be familiar with and use the symbols shown in *Appendix 2: Symbols.*

Content

Topic 1: Computer systems

1.1 Computer architecture	
Content	**Learners should:**
1.1.1 Stored-program concept	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks
1.1.2 von Neumann architecture	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks
1.1.3 Internal components of a computer system: **a.** Central processing unit (CPU) **b.** Memory: • Random access memory (RAM) • Read-only memory (ROM) • Cache **c.** Input/output **d.** Buses: • Address • Data • Control	Know and understand: • Definition • Function • Characteristics • Why needed • Operation Be able to: • interpret and complete a diagrammatic representation of the internal components of a computer system based on von Neumann architecture • develop an expression for the amount of addressable memory.

1.1 Computer architecture *(continued)*

Content	Learners should:
1.1.4 Components of the CPU: **a.** Control unit (CU) **b.** Arithmetic logic unit (ALU) **c.** General-purpose registers **d.** Special-purpose registers: • Program counter (PC) • Memory address register (MAR) • Memory buffer register (MBR) • Current instruction register (CIR) • Accumulator (ACC) • Stack pointer (SP) • Status register (SR) **e.** Clock	Know and understand: • Definition • Function • Characteristics • Why needed • Operation Be able to: • interpret and complete a diagrammatic representation of the components of the CPU. Learners should be familiar with and be able to use the CPU component symbols shown in *Appendix 2: Symbols.*
1.1.5 Fetch-decode-execute (FDE) cycle: **a.** Stages **b.** Role of components **c.** Use of read/write control signals	Know and understand: • Definition • Function • Why needed • Operation Be able to: • use register transfer notation to show how data passes between registers during the FDE cycle • interpret and complete a diagrammatic representation of the transfer of data between components during the FDE cycle. Learners should be familiar with and be able to use the register transfer notation shown in *Appendix 2: Symbols.*

1.1 Computer architecture *(continued)*

Content	Learners should:
1.1.6 Memory hierarchy: **a.** CPU registers **b.** Cache: • Level 1 • Level 2 • Level 3 **c.** Main memory **d.** Secondary storage	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of the memory hierarchy.
1.1.7 Data transmission: **a.** Serial **b.** Parallel	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of data transmission.

1.2 The operating system (OS)	
Content	**Learners should:**
1.2.1 Role of the OS: **a.** User interface **b.** User management **c.** Peripheral management **d.** Process management **e.** Memory management	Know and understand: • Definition • Function • Characteristics • Why needed
1.2.2 Multitasking: **a.** Time sharing **b.** Context switching **c.** Interrupt handling in multitasking	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of the interrupt handling process.

1.2 The operating system (OS) *(continued)*	
Content	**Learners should:**
1.2.3 Process management: **a.** Process **b.** Process states **c.** Scheduling **d.** Scheduling algorithms: • Round robin (RR) • First-come, first-served (FCFS) • Shortest job first (SJF) • Multi-level queue	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of process states • interpret and complete a diagrammatic representation of a scheduling algorithm.
1.2.4 Memory management: **a.** Operations: • Tracking memory allocation status • Determining storage requirements • Controlling memory usage **b.** Allocation techniques: • Paging • Segmentation • Virtual memory **c.** Stack frame	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of memory usage • interpret and complete a diagrammatic representation of the contents of a call stack.

Content

Topic 2: Data representation

2.1 Numbers	
Content	**Learners should:**
2.1.1 Number bases: **a.** Binary **b.** Denary **c.** Hexadecimal	Know and understand: • Definition • Function • Characteristics • Why needed Be able to: • convert unsigned integers between bases.
2.1.2 Units of measurement: **a.** Binary numbers: • bit (b) • nibble • byte (B) **b.** IEC binary prefixes: • kibi (Ki) • mebi (Mi) • gibi (Gi) • tebi (Ti) **c.** SI decimal prefixes: • kilo (K) • mega (M) • giga (G) • tera (T)	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • use binary units of measurement to express data capacity and file sizes • use decimal units of measurement to denote transmission speeds • rank units of measurement in size order • convert between units of measurement • interpret and write expressions using units of measurement • calculate the maximum number of states that can be represented by a given number of bits.

2.1 Numbers *(continued)*	
Content	**Learners should:**
2.1.3 Two's complement representation of signed numbers	Know and understand: • Definition • Function • Characteristics • Why needed • Operation Be able to: • convert signed denary integers into two's complement signed binary and vice versa • use negation to convert from one sign to another.
2.1.4 Fixed-point representation of signed fractional numbers	Know and understand: • Definition • Function • Characteristics • Why needed • Operation Be able to: • convert signed fractional denary numbers to fixed-point signed binary and vice versa • work with halves, quarters and eighths.

2.2 Binary arithmetic

Content	Learners should:
2.2.1 Addition and subtraction	Know and understand: • Definition • Function • Operation Be able to: • perform addition and subtraction on signed and unsigned binary numbers.
2.2.2 Bitwise manipulation: **a.** Logical shift **b.** Arithmetic shift **c.** Bit masks: • AND • OR • XOR	Know and understand: • Definition • Function • Characteristics • Why needed • Operation Be able to: • perform binary shifts • use a mask to check the status of bits in a binary pattern.
2.2.3 Overflow and underflow	Know and understand: • Definition • Function • Characteristics

2.3 Text

Content	Learners should:
2.3.1 Character encoding systems: **a.** ASCII/extended ASCII **b.** UTF-8/Unicode	Know and understand: • Definition • Function • Characteristics • Why needed • Benefits • Drawbacks

Content

Topic 3: Networks and encryption

3.1 Network fundamentals	
Content	**Learners should:**
3.1.1 Network components: **a.** Network interface card (NIC) **b.** Switch **c.** Home router **d.** Modem **e.** Wireless access point (WAP) **f.** Transmission media: • Copper • Wireless • Fibre **g.** Firewall • Hardware • Software	Know and understand: • Definition • Function • Why needed • Characteristics • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of a network.
3.1.2 Network components: **a.** Bandwidth **b.** Speed/transfer rate **c.** Latency **d.** Range	Know and understand: • Definition • Characteristics Be able to: • develop expressions involving file size, transfer rate and time.
3.1.3 Network addressing: **a.** Internet protocol (IP): • IPv4 • IPv6 **b.** Media access control (MAC)	Know and understand: • Definition • Function • Characteristics • Why needed • Operation Be able to: • develop expressions involving network addresses.

3.2 Encryption

Content	Learners should:
3.2.1 Encryption fundamentals	Know and understand: • Definition • Function • Characteristics • Why needed • Benefits • Drawbacks
3.2.2 Methods of encryption: **a.** Symmetric: • Known shared key • Key exchange **b.** Asymmetric: • Public/private key pair	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of the encryption/ decryption process.
3.2.3 Monoalphabetic substitution cipher	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • encrypt a plaintext message/ decrypt a ciphertext message using a monoalphabetic substitution cipher.

3.2 Encryption *(continued)*	
Content	**Learners should:**
3.2.4 Vernam cipher algorithm	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • encrypt a plaintext message/decrypt a ciphertext message using a Vernam cipher.

Content

Topic 4: Structuring data

4.1 Relational databases	
Content	**Learners should:**
4.1.1 Entity-relationship model (ERM): **a.** Entity **b.** Attribute **c.** Key: • Primary • Foreign • Composite **d.** Relationship cardinality: • One-to-one (1:1) • One-to-many (1:M) • Many-to-many (M:M) **e.** Ways of representing: • Entity-relationship diagram (ERD) • Parenthetical notation	Know and understand: • Definition • Function • Characteristics • Why needed • Operation Be able to: • interpret, complete, find and correct errors in an ERD • interpret, complete, find and correct errors in a schema expressed in parenthetical notation. Learners should be familiar with and be able to use the ERD symbols shown in *Appendix 2: Symbols*. Learners' understanding of relational databases is assessed in Unit 1. Their ability to connect to and use an SQLite database is assessed in Unit 2.
4.1.2 Logical schema: **a.** Table/relation **b.** Column/attribute **c.** Row/tuple **d.** Data types **e.** Integrity constraints: • Entity • Referential • Domain	Know and understand: • Definition • Function • Characteristics • Why needed • Operation Learners' understanding of relational databases is assessed in Unit 1. Their ability to connect to and use an SQLite database is assessed in Unit 2.

4.2 Data structures	
Content	**Learners should:**
4.2.1 Data structures: **a.** Array **b.** List **c.** Dictionary **d.** Record **e.** Tuple **f.** Set	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • hand trace operations on one- and two-dimensional data structures • show the contents of a data structure after operations have been performed • perform common operations on sets. Learners' understanding of data structures is assessed in Unit 1. Their ability to create and handle data structures in code is assessed in Unit 2.

4.3 Abstract data types (ADTs)	
Content	**Learners should:**
4.3.1 Stack: **a.** Structure **b.** Operations: • Insert • Delete • Search	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • hand trace operations on a stack • show the contents of a stack after operations have been performed. Learners' understanding of the stack ADT is assessed in Unit 1. Their ability to represent and handle a stack implemented as a fixed-length list is assessed in Unit 2.
4.3.2 Queue: **a.** Structure **b.** Operations: • Insert • Delete • Search	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • hand trace operations on a queue • show the contents of a queue after operations have been performed. Learners' understanding of the queue ADT is assessed in Unit 1. Their ability to represent and handle a queue implemented as a fixed-length list is assessed in Unit 2.

Content

Topic 5: Problem solving

5.1 Tools and techniques	
Content	**Learners should:**
5.1.1 Algorithms	Know and understand: • Definition • Function • Characteristics • Why needed • Operation Be able to: • interpret, complete and develop algorithms expressed in words, diagrams, flowcharts and code • analyse the efficiency of algorithms • find and correct logic errors in algorithms. Learners should be familiar with and be able to use the flowchart symbols shown in *Appendix 2: Symbols*. Learners' ability to interpret and develop algorithms is assessed in Unit 1. Their ability to convert algorithms into code is assessed in Unit 2.
5.1.2 Trace table	Know and understand: • Definition • Function • Characteristics • Why needed • Operation Be able to: • complete a trace table for an algorithm.

5.1 Tools and techniques *(continued)*	
Content	**Learners should:**
5.1.3 Recursion	Know and understand: • Definition • Function • Characteristics • Why needed • Operation Be able to: • hand trace the operation of a recursive algorithm. Learners' understanding of recursion is assessed in Unit 1. Their ability to use recursion in code is assessed in Unit 2.

5.2 Searching and sorting algorithms	
Content	**Learners should:**
5.2.1 Linear search algorithm: **a.** Unsorted array **b.** Sorted array	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • hand trace the operation of a linear search algorithm. Learners' understanding of how a linear search algorithm works is assessed in Unit 1. Their ability to implement a linear search in code is assessed in Unit 2.

5.2 Searching and sorting algorithms *(continued)*	
Content	**Learners should:**
5.2.2 Binary search algorithm: **a.** Iterative **b.** Recursive	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • hand trace the operation of a binary search algorithm. Learners' understanding of how a binary search algorithm works is assessed in Unit 1. Their ability to implement a binary search in code is assessed in Unit 2.
5.2.3 Bubble sort algorithm	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • hand trace the operation of a bubble sort algorithm. Learners' understanding of how a bubble sort algorithm works is assessed in Unit 1. Their ability to implement a bubble sort in code is assessed in Unit 2.

5.2 Searching and sorting algorithms *(continued)*	
Content	**Learners should:**
5.2.4 Insertion sort algorithm	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • hand trace the operation of an insertion sort algorithm. Learners' understanding of how an insertion sort algorithm works is assessed in Unit 1. Their ability to implement an insertion sort in code is assessed in Unit 2.

5.3 Boolean logic	
Content	**Learners should:**
5.3.1 Boolean expressions: **a.** Boolean operators: • AND • OR • XOR • NOT • NAND • NOR **b.** Order of precedence	Know and understand: • Definition • Function • Why needed • Operation Be able to: • interpret and create Boolean expressions. Learners should be familiar with and be able to use the Boolean algebraic notation shown in *Appendix 2: Symbols*.
5.3.2 Truth table	Know and understand: • Definition • Function • Why needed • Operation Be able to: • interpret and complete a truth table for a given logical expression.

5.4 Programming paradigms	
Content	**Learners should:**
5.4.1 Procedural programming paradigm	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Learners' understanding of the procedural programming paradigm is assessed in Unit 1. Their ability to use procedural programming is assessed in Unit 2.
5.4.2 Declarative programming paradigm	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Learners' understanding of the declarative programming paradigm is assessed in Unit 1. Their ability to use declarative programming is assessed in Unit 2.

Content

Topic 6: Enabling technologies

6.1 Data science	
Content	**Learners should:**
6.1.1 Big data: **a.** Five Vs (volume, velocity, variety, veracity, value) **b.** Infrastructure: • Collection • Storage • Processing • Modelling and analysis • Networking • Security	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of a big data system.
6.1.2 Tools and techniques of data science: **a.** Collection **b.** Storage **c.** Web scraping **d.** Cleaning **e.** Analysis **f.** Visualisation	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks

6.2 Artificial intelligence (AI)

Content	Learners should:
6.2.1 Categories of AI: **a.** Narrow **b.** General **c.** Super	Know and understand: • Definition • Characteristics • Why needed • Benefits • Drawbacks
6.2.2 Machine learning (ML): **a.** Supervised **b.** Unsupervised	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of a supervised/ unsupervised ML system.
6.2.3 Large language model (LLM): **a.** Context and query accuracy **b.** Knowledge limitations **c.** Bias/subjectivity	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of a LLM system.

6.3 Tools

Content	Learners should:
6.3.1 Integrated development environment (IDE): **a.** Text editor **b.** Autocomplete **c.** Syntax checker **d.** Debugger: • Breakpoints • Stepping **e.** Memory inspection	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Learners' understanding of features of an IDE is assessed in Unit 1. They are expected to use an IDE when writing program code in Unit 2.
6.3.2 Version control system (VCS)	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Learners are not expected to have experience of using a VCS.
6.3.3 Open-source technologies	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks

6.3 Tools *(continued)*	
Content	**Learners should:**
6.3.4 Application programming interface (API)	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Learners' understanding of APIs is assessed in Unit 1. They are expected to use an API when working with SQLite databases in Unit 2.

Unit 2: Practical Programming and Problem-solving

Assessment information

First assessment: May/June 2027

The assessment is 3 hours.

The assessment consists of five compulsory questions.

The assessment is out of 80 marks.

Learners must answer all questions.

The questions are practical in nature and require learners to design, write, test and refine programs in order to solve problems.

Learners must complete the assessment onscreen using an integrated development environment (IDE) of choice.

Learners must answer the questions onscreen using Python 3.

Learners will be provided with a hard copy of the question paper, digital coding files and digital Programming Language Subset (PLS) document.

Centres must ensure secure examination conditions. For more information refer to the section: *Assessment information Unit 2 and Unit 4 practical programming.*

Sample assessment materials

A sample paper and mark scheme for this unit can be found in the *Pearson Edexcel International Advanced Level in Computer Science Sample Assessment Materials (SAMs)* document.

A full list of command words that will be used in assessments is shown in *Appendix 5: Command word taxonomy.*

Overview

Learners apply computational thinking and programming knowledge in a practical, onscreen environment using Python 3. They will design, write, test, and refine solutions to realistic problems, employing best practices in code development. Key focus areas include structured programming, data handling, modular design and algorithmic thinking. This unit enhances learners' ability to produce efficient, maintainable code and to reason logically when developing software solutions.

The theory of programming covered in Topics 4 and 5 of Unit 1 provides a solid foundation for the practical work learners undertake in Unit 2.

Learners are expected to be familiar with and use the Programming Language Subset (PLS) document provided on the *Pearson Qualifications Website in International Advanced Levels in Computer Science* and the symbols shown in *Appendix 2: Symbols.*

Content

Topic 7: Programming

7.1 Constructs	
Content	**Learners should:**
7.1.1 Flow control constructs: **a.** Sequence **b.** Selection **c.** Count-controlled iteration **d.** Condition-controlled iteration **e.** Exception handling for built-in exceptions	Be able to: • select and use appropriate flow control constructs. Further details provided in the Programming Language Subset (PLS) document.
7.1.2 Variables and constants: **a.** Creation **b.** Assignment **c.** Access	Be able to: • select and use appropriate variables and constants. Further details provided in the Programming Language Subset (PLS) document.
7.1.3 Operators: **a.** Arithmetic **b.** Relational **c.** Boolean **d.** Bitwise	Be able to: • select and use appropriate operators. Further details provided in the Programming Language Subset (PLS) document.
7.1.4 Subprograms: **a.** Parameter **b.** Argument **c.** Function **d.** Procedure	Be able to: • select and use appropriate subprograms. Further details provided in the Programming Language Subset (PLS) document.

Content

Topic 8: Organising and handling data

8.1 Data types and structures	
Content	**Learners should:**
8.1.1 Data types: **a.** Integer **b.** Real **c.** Character **d.** Boolean **e.** String	Be able to: • select and use appropriate data types • convert between data types. Further details provided in the Programming Language Subset (PLS) document.
8.1.2 Data structures: **a.** Array **b.** List **c.** Dictionary **d.** Record **e.** Tuple **f.** Set	Be able to: • select and use one- and two-dimensional data structures • select and use one- and two-dimensional lists as equivalent to one- and two-dimensional arrays • convert between data structures. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of data structures is assessed in Unit 1. Their ability to create and handle data structures in code is assessed in Unit 2.

8.2 Data handling methods	
Content	**Learners should:**
8.2.1 Methods to handle numeric data: a. Rounding b. Truncation c. Randomisation d. Data type conversions	Be able to: • select and use appropriate methods for handling numeric data. Further details provided in the Programming Language Subset (PLS) document.
8.2.2 Methods to handle string data: a. Length b. Indexing c. Formatting d. Examining e. Manipulating	Be able to: • select and use appropriate methods for handling string data. Further details provided in the Programming Language Subset (PLS) document.
8.2.3 Methods to handle text files: a. Open b. Read from c. Write to d. Append to e. Close	Be able to: • select and use appropriate methods for handling text files. Further details provided in the Programming Language Subset (PLS) document.
8.2.4 Methods to handle data structures: a. Create b. Insert c. Update d. Retrieve e. Delete f. Traverse	Be able to: • select and use appropriate methods for handling data structures. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of data structures is assessed in Unit 1. Their ability to create and handle data structures in code is assessed in Unit 2.

8.2 Data handling methods *(continued)*	
Content	**Learners should:**
8.2.5 Methods to represent and handle a stack implemented as a fixed-length list: **a.** Create **b.** Push **c.** Pop **d.** IsEmpty **e.** IsFull **f.** Size	Be able to: • select and use appropriate methods to handle a stack implemented as a fixed-length list. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of the stack ADT is assessed in Unit 1. Their ability to create and handle a stack implemented as a fixed-length list is assessed in Unit 2.
8.2.6 Methods to represent and handle a queue implemented as a fixed-length list: **a.** Create **b.** Enqueue **c.** Dequeue **d.** IsEmpty **e.** IsFull **f.** Size	Be able to: • select and use appropriate methods to handle a queue implemented as a fixed-length list. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of the queue ADT is assessed in Unit 1. Their ability to create and handle a queue implemented as a fixed-length list is assessed in Unit 2.
8.2.7 Methods to handle a set: **a.** Membership **b.** Union **c.** Intersection **d.** Difference	Be able to: • select and use appropriate methods to handle a set. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of sets is assessed in Unit 1. Their ability to use set operations in code is assessed in Unit 2.

8.3 Relational databases

Content	Learners should:
8.3.1 Methods to administer an SQLite database	Be able to: • select and use appropriate methods to administer an SQLite database. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of relational databases is assessed in Unit 1. Their ability to administer an SQLite database is assessed in Unit 2.
8.3.2 Methods to manipulate data from an SQLite database	Be able to: • select and use appropriate methods to manipulate data from an SQLite database. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of relational databases is assessed in Unit 1. Their ability to manipulate an SQLite database is assessed in Unit 2.
8.3.3 Methods to retrieve data from an SQLite database	Be able to: • select and use appropriate methods to retrieve data from an SQLite databases. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of relational databases is assessed in Unit 1. Their ability to query an SQLite database is assessed in Unit 2.

8.3 Relational databases *(continued)*	
Content	**Learners should:**
8.3.4 Methods to filter and refine data retrieved from an SQLite database	Be able to: • select and use appropriate methods to filter and refine data retrieved from an SQLite database. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of relational databases is assessed in Unit 1. Their ability to filter and refine data retrieved from an SQLite database is assessed in Unit 2.

Content

Topic 9: Best practice

9.1 Program design	
Content	**Learners should:**
9.1.1 Methods to produce correct outcomes: **a.** Problem decomposition into component parts **b.** Clear and appropriate logic **c.** Variables, constants, data types and data structures appropriate to the problem **d.** Programming constructs appropriate to the problem	Be able to: • select and use appropriate methods to produce correct outcomes. Further details provided in the Good Programming Practice Guide (GPPG).
9.1.2 Methods to implement good practice in program design: **a.** Modularity **b.** Separation of concerns **c.** Minimisation of side effects **d.** Scope isolation **e.** Fitness for purpose of input and output **f.** Efficient solutions in terms of execution time and memory usage	Be able to: • select and use appropriate methods to design effective code. Further details provided in the Good Programming Practice Guide (GPPG).
9.1.3 Methods to implement modularity in source code: **a.** Blocked code **b.** User-defined subprograms **c.** Built-in subprograms **d.** Libraries of subprograms **e.** Files **f.** Procedures **g.** Functions	Be able to: • select and use appropriate methods to produce modular code. Further details provided in the Programming Language Subset (PLS) document and Good Programming Practice Guide (GPPG).

9.2 Clean code

Content	Learners should:
9.2.1 Consistency: **a.** Use of white space **b.** Line breaks layout **c.** Naming conventions **d.** Fixed line length	Be able to: • select and use appropriate methods to improve the consistency of code. Further details provided in the Good Programming Practice Guide (GPPG).
9.2.2 Readability: **a.** Meaningful identifiers **b.** Informative comments **c.** Named constants **d.** Subprograms	Be able to: • select and use appropriate methods to improve the readability of code. Further details provided in the Good Programming Practice Guide (GPPG).
9.2.3 Methods to control the behaviour of program code: **a.** Boundary value analysis **b.** Exception handling **c.** Minimising nested conditionals **d.** Minimising loops **e.** Using guard clauses **f.** Managed conversion of data types	Be able to: • select and use appropriate methods to control the behaviour of code. Further details provided in the Programming Language Subset (PLS) document and Good Programming Practice Guide (GPPG).

9.3 Functionality

Content	Learners should:
9.3.1 Methods to ensure that code is fit for purpose, produces accurate results and meets requirements: **a.** Testing and debugging **b.** Reviewing and refining	Be able to: • identify errors in program code • correct errors in program code • produce accurate outcomes • refine solutions to meet requirements. Further details provided in the Good Programming Practice Guide (GPPG).

Content

Topic 10: Computational thinking and algorithms

10.1 Computational thinking	
Content	**Learners should:**
10.1.1 Abstraction	Be able to: • use subprograms to hide implementation details.
10.1.2 Decomposition	Be able to: • use comment-first coding to describe an algorithm.
10.1.3 Pattern recognition	Be able to: • use the same or similar solution/algorithm/subprogram in a different problem • use the same or similar data structure in a different problem.
10.1.4 Generalisation	Be able to: • use a solution/algorithm/ subprogram for similar problems by changing input parameters.

10.2 Implementing algorithms	
Content	**Learners should:**
10.2.1 Create programmed solutions from algorithms: **a.** Expressed in words **b.** Expressed in diagrams **c.** Expressed in flowcharts	Be able to: • convert algorithms into code.

10.3 Search and sort algorithms

Content	Learners should:
10.3.1 Linear search algorithm: **a.** Worst-case on unsorted list **b.** Early exit when found on unsorted list **c.** Early exit when found on sorted list **d.** Early exit when not in list on sorted list	Be able to: • implement and use an appropriate linear search algorithm. Learners' understanding of how a linear search algorithm works is assessed in Unit 1. Their ability to amend and write a linear search in code is assessed in Unit 2.
10.3.2 Binary search algorithm: **a.** Iterative **b.** Recursive	Be able to: • implement and use an appropriate binary search algorithm. Learners' understanding of how a binary search algorithm works is assessed in Unit 1. Their ability to amend and write a binary search in code is assessed in Unit 2.
10.3.3 Bubble sort algorithm: **a.** Worst case **b.** Exit with no swaps **c.** Reduce upper bound on each pass	Be able to: • implement and use an appropriate bubble sort algorithm. Learners' understanding of how a bubble sort algorithm works is assessed in Unit 1. Their ability to amend and write a bubble sort in code is assessed in Unit 2.
10.3.4 In-place insertion sort algorithm	Be able to: • implement and use an in-place insertion sort algorithm. Learners' understanding of how an in-place insertion sort algorithm works is assessed in Unit 1. Their ability to amend and write an insertion sort in code is assessed in Unit 2.

10.4 Recursion

Content	Learners should:
10.4.1 Recursive algorithms	Be able to: • implement and use recursive algorithms. Learners' understanding of recursion is assessed in Unit 1. Their ability to amend and write recursion in code is assessed in Unit 2.

Unit 3: Advanced Principles of Computer Science

Assessment information

First assessment: May/June 2028

The assessment is 1 hours and 30 minutes.

The assessment is out of 80 marks.

Learners must answer all questions.

This unit may include multiple choice, short/medium open response, extended open response, tabular and diagrammatic items.

Calculators must not be used in the examination.

Learners will be provided with the assembly language instruction set shown in *Appendix 3: Assembly language instruction set.*

Sample assessment materials

A sample paper and mark scheme for this unit can be found in the *Pearson Edexcel International Advanced Level in Computer Science Sample Assessment Materials (SAMs) document.*

A full list of command words that will be used in assessments is shown in *Appendix 5: Command word taxonomy.*

Overview

This unit builds on the knowledge gained in Unit 1 and extends learners' theoretical understanding of advanced computing concepts. Learners study complex systems architecture, data representation methods, networking models, cybersecurity and the use of multiple programming paradigms. Additional emphasis is placed on emerging technologies and the professional, legal and ethical issues surrounding their use. Learners will analyse and evaluate technologies and approaches, strengthening their analytical and critical thinking skills.

Topics 13, 14 and 15 deliver the advanced programming theory that learners need in order to tackle the practical coding which is the focus of Unit 4.

Learners are expected to be familiar with and use the assembly language instruction set shown in *Appendix 3: Assembly language instruction set*, and the symbols shown in *Appendix 2: Symbols*.

Content

Topic 11: Computer systems and data representation

11.1 Computer architecture	
Content	**Learners should:**
11.1.1 Harvard architecture: **a.** Memory structure **b.** Difference between Harvard and von Neumann architecture	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of the internal components of a computer system based on the Harvard architecture. Learners should be familiar with and be able to use the CPU component symbols shown in *Appendix 2: Symbols*.
11.1.2 Complex instruction set (CISC) and reduced instruction set computer (RISC): **a.** Architecture **b.** Difference between RISC and CISC	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks
11.1.3 Five-stage instruction pipelining: **a.** Stages: • Instruction fetch (IF) • Instruction decode (ID) • Execute (EX) • Memory Access (MEM) • Register write back (WB) **b.** Branch hazards	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of the pipelining process.

11.1 Computer architecture *(continued)*	
Content	**Learners should:**
11.1.4 Parallel processing: **a.** Multicore processors: • Multiple instruction multiple data (MIMD) **b.** Graphic processing units (GPUs): • Single instruction multiple data (SIMD)	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • identify and complete a diagrammatic representation of parallel processing.
11.1.5 Embedded systems: **a.** Hardware components: • Microprocessor • Microcontroller unit (MCU) • Memory • I/O interfaces and ports • Sensors: o Accelerometer o Vibration o Sound o Temperature • Actuators • Analogue-digital converter (ADC/DAC): o Sampling o Quantisation o Encoding • Power supply **b.** Software components: • Firmware • Operating system (OS) • Application software • Programming languages used to develop software for embedded systems	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • interpret and complete a design for an embedded system • interpret and complete a diagrammatic representation of the ADC/DAC process.

11.2 The operating system	
Content	**Learners should:**
11.2.1 Input/output (I/O) device management: **a.** Role of device drivers **b.** Basic I/O system (BIOS) **c.** Role of buffering **d.** Role of caching **e.** Interrupt handling in device management	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of I/O in a computer system.
11.2.2 Virtualisation: **a.** Containerisation **b.** Virtual machines	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of a containerised system. • interpret and complete a diagrammatic representation of a virtual machine.

11.3 Data representation	
Content	**Learners should:**
11.3.1 Floating-point representation (IEEE-754)	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • convert fractional numbers into binary and vice versa using 16–bit half precision IEEE-754 standard (1–bit sign), 5–bit exponent (biased), and 10–bit significand.

Content

Topic 12: Networks and cybersecurity

12.1 Networking	
Content	**Learners should:**
12.1.1 Open systems interconnection (OSI) reference model: **a.** Layers **b.** Encapsulation/decapsulation	Know and understand: • Definition • Function • Characteristics • Why needed • Operation Be able to: • interpret and complete a diagrammatic representation of encapsulation/ decapsulation of packets flowing through the OSI protocol stack.
12.1.2 Protocol stacks: **a.** TCP/IP **b.** SSL/TLS	Know and understand: • Definition • Function • Characteristics • Why needed • Operation Be able to: • interpret and complete a diagrammatic representation of encapsulation/ decapsulation of packets flowing through the TCP/IP protocol stack • interpret and complete a diagrammatic representation of the SSL/TLS handshake.

12.1 Networking *(continued)*	
Content	**Learners should:**
12.1.3 Packet switching: **a.** Packet components **b.** Internet router **c.** Protocols	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Learners are not required to study circuit switching.
12.1.4 Domain name system (DNS)	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of how the DNS works.

12.2 The Internet of Things (IoT)	
Content	**Learners should:**
12.2.1 The IoT stack: **a.** Device hardware **b.** Device software: • Operating system • Applications **c.** Connectivity: • Networks • Protocols **d.** Cloud platform: • Data collection • Analytics • APIs **e.** Cloud applications	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of an IoT system.

12.3 Cybersecurity

Content	Learners should:
12.3.1 CIA triad model	Know and understand: • Definition • Characteristics • Why needed
12.3.2 RSA encryption algorithm: **a.** Prime numbers **b.** Key exchange **c.** Vulnerabilities	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of RSA encryption.
12.3.3 Error detection: **a.** Parity bit **b.** Two-dimensional parity block **c.** Checksum	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • calculate the parity bit for a given set of bits • calculate the parity bits for a given two-dimensional parity block • calculate the checksum for a given block of data.

12.4 Computing paradigms

Content	Learners should:
12.4.1 Cloud computing: **a.** Infrastructure **b.** Services	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks
12.4.2 Edge computing: **a.** Infrastructure **b.** Services	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks

Content

Topic 13: Programming languages

13.1 Types of programming languages	
Content	**Learners should:**
13.1.1 Object-oriented programming (OOP) paradigm: **a.** Concepts: • Abstraction • Class • Instantiation • Methods and attributes (public, private, protected) • Encapsulation • Composition • Inheritance • Polymorphism **b.** Unified modelling language (UML) class diagrams	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • interpret and complete UML class diagrams • identify object-oriented programming (OOP) features implemented in code. Learners should be familiar with and be able to use UML class notation shown in *Appendix 2: Symbols*. Learners' understanding of the object-oriented programming (OOP) paradigm is assessed in Unit 3. Their ability to use object-oriented programming (OOP) is assessed in Unit 4.
13.1.2 Functional programming paradigm: **a.** Concepts: • Pure functions • First-class functions • Higher-order functions • Recursion • Immutable variables • Avoid side-effects	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Learners' understanding of the functional programming paradigm is assessed in Unit 3. Their ability to use functional programming is assessed in Unit 4.

13.1 Types of programming languages *(continued)*	
Content	**Learners should:**
13.1.3 Assembly language: a. Mnemonics b. Instructions: • Opcode • Operand c. Addressing modes: • Immediate • Direct d. Operations: • Arithmetic • Bitwise • Branch • Data movement • Halt e. Comments	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • read and write simple assembly language code • trace the execution of a simple assembly language program. Learners should be familiar with and able to use the assembly language instructions and operations shown in *Appendix 3: Assembly language instruction set.*

Content

Topic 14: Structuring data

14.1 Abstract data types (ADTs)	
Content	**Learners should:**
14.1.1 Linked list: **a.** Structure **b.** Ways to represent: • Classes • References **c.** Operations: • Insert • Delete • Search • Traverse	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • hand trace operations on a linked list • show the contents of a linked list after operations have been performed. Learners' understanding of the linked list ADT is assessed in Unit 3. Their ability to implement and handle a linked list is assessed in Unit 4.
14.1.2 Hash table: **a.** Structure **b.** Ways to represent: • Classes • References **c.** Operations: • Insert • Delete • Search • Traverse • Calculate hash values • Handle collisions • Load factor and rehashing	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • hand trace operations on a hash table • show the contents of a hash table after operations have been performed • calculate hash values and handle collisions. Learners' understanding of the hash table ADT is assessed in Unit 3. Their ability to implement and handle a hash table is assessed in Unit 4.

14.1 Abstract data types (ADTs) *(continued)*	
Content	**Learners should:**
14.1.3 Graph: **a.** Structure **b.** Ways to represent: • Adjacency matrix • Adjacency list **c.** Types: • Undirected • Directed • Weighted **d.** Operations: • Insert • Delete • Search • Traverse [breath, depth]	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • hand trace operations on a graph • show the contents of a graph after operations have been performed. Learners' understanding of the graph ADT is assessed in Unit 3. Their ability to implement and handle a graph is assessed in Unit 4.
14.1.4 Tree: **a.** Structure **b.** Types: • Binary tree • Binary search tree **c.** Operations: • Insert • Delete • Search • Traverse [pre-order, in-order, post-order]	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • hand trace operations on a tree • show the contents of a tree after operations are performed. Learners' understanding of the tree ADT is assessed in Unit 3. Their ability to implement and handle a tree is assessed in Unit 4.

14.1 Abstract data types (ADTs) *(continued)*	
Content	**Learners should:**
14.1.5 Circular queue: **a.** Structure **b.** Operations: • Insert • Delete • Search	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • hand trace operations on a circular queue • show the contents of a circular queue after operations are performed. Learners' understanding of the circular queue ADT is assessed in Unit 3. Their ability to implement and use a circular queue implemented as a fixed-length list is assessed in Unit 4.

Content

Topic 15: Problem solving

15.1 Algorithm design	
Content	**Learners should:**
15.1.1 Problem-solving techniques: **a.** Brute force/exhaustive **b.** Divide and conquer **c.** Greedy algorithms **d.** Heuristics **e.** Backtracking **f.** Recursion **g.** Tracing algorithms	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • complete a trace table for an algorithm. Learners' understanding of problem-solving techniques is assessed in Unit 3. Their ability to use them when writing code is assessed in Unit 4.

15.2 Algorithms	
Content	Learners should:
15.2.1 Merge sort algorithm	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • hand trace the operation of a merge sort algorithm. Learners' understanding of how the merge sort algorithm works is assessed in Unit 3. Their ability to implement a recursive merge sort in code is assessed in Unit 4.

15.2 Algorithms *(continued)*

Content	**Learners should:**
15.2.2 Quick sort algorithm	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • hand trace the operation of a quick sort algorithm. Learners' understanding of how the quick sort algorithm works is assessed in Unit 3. Their ability to implement a recursive quick sort in code is assessed in Unit 4.
15.2.3 Shortest path algorithms: **a.** Dijkstra's algorithm **b.** A* algorithm	Know and understand: • Definition • Function • Characteristics • Operation • Benefits • Drawbacks Be able to: • hand trace the operation of a Dijkstra's shortest path algorithm. Learners' understanding of how the Dijkstra's shortest path algorithm works is assessed in Unit 3. Their ability to implement the algorithm in code is assessed in Unit 4.

15.3 Algorithmic efficiency

Content	Learners should:
15.3.1 Big O notation: **a.** Time complexity: • Constant • Logarithmic • Linear • Linearithmic • Polynomial • Exponential	Know and understand: • Definition • Function • Why needed Be able to: • analyse algorithms to establish best and worst Big O • give the best and worst Big O time complexity for common algorithms • determine the Big O time complexity for given code.

15.4 Boolean logic

Content	Learners should:
15.4.1 Boolean algebra: **a.** Commutative **b.** Associative **c.** Absorption **d.** Distributive **e.** De Morgan's **f.** Double negation	Know and understand: • Definition • Function • Why needed • Operation Be able to: • use Boolean algebra to manipulate Boolean expressions such as demonstrating that one expression is equivalent to another. Learners should be familiar with and be able to use the Boolean algebraic notation shown in *Appendix 2: Symbols*.
15.4.2 Karnaugh map	Know and understand: • Definition • Function • Why needed • Operation • Benefits • Drawbacks Be able to: • use Karnaugh maps to simplify Boolean algebraic expressions.

Content

Topic 16: Emerging technologies and professional practice

16.1 Emerging technologies	
Content	**Learners should:**
16.1.1 Encryption: **a.** Digital signatures and certificates **b.** Hash digest and use of public/private keys	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks
16.1.2 Blockchain	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks Be able to: • interpret and complete a diagrammatic representation of a blockchain.
16.1.3 Quantum computing: **a.** Qubits **b.** Principles • Superposition • Entanglement • Decoherence • Interference **c.** Threat **d.** Cryptography	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks

16.1 Emerging technologies *(continued)*	
Content	**Learners should:**
16.1.4 Deep learning: **a.** Neural networks: • Layers • Forward propagation • Back propagation • Weights • Bias • Activation function	Know and understand: • Definition • Function • Characteristics • Why needed • Operation • Benefits • Drawbacks

16.2 Professionalism	
Content	**Learners should:**
16.2.1 Responsible practice: **a.** Ethical issues **b.** Legal frameworks **c.** Codes of conduct	Know and understand: • Definition • Function • Characteristics • Why needed

Unit 4: Advanced Practical Programming and Problem-solving

Assessment information

First assessment: May/June 2028

The assessment is 3 hours.

The assessment consists of five compulsory questions.

The assessment is out of 80 marks.

Learners must answer all questions.

The questions are practical in nature and require learners to design, write, test and refine programs in order to solve problems.

Learners must complete the assessment onscreen using an integrated development environment (IDE) of choice.

Learners must answer the questions onscreen using Python 3.

Learners will be provided with a hard copy of the question paper, digital coding files and digital Programming Language Subset (PLS) document.

Centres must ensure secure examination conditions. For more information refer to the section: *Assessment information - Unit 2 and Unit 4 practical programming.*

Sample assessment materials

A sample paper and mark scheme for this unit can be found in the *Pearson Edexcel International Advanced Level in Computer Science Sample Assessment Materials (SAMs) document.*

A full list of command words that will be used in assessments is shown in *Appendix 5: Command word taxonomy.*

Overview

In this advanced practical unit, learners design and implement robust software solutions, applying the core concepts and best practice techniques learnt in Unit 4. They demonstrate skills in object-oriented and functional programming, work with abstract data types, and apply advanced algorithms such as pathfinding and recursive sorting. The unit also focuses on best practices, such as modularity, documentation and testing strategies, preparing learners for further study and professional coding environments.

There is some overlap with Unit 2, which highlights that although the problems that learners tackle become more complex, the approaches and tools they use to solve them remain consistent.

The advanced theory of programming covered in Topics 13, 14 and 15 of Unit 3 underpins and supports the practical coding, which is the focus of Unit 4.

Learners are expected to be familiar with and use the Programming Language Subset (PLS) document provided on the *Pearson Qualifications Website in International Advanced Levels in Computer Science* and the symbols shown in *Appendix 2: Symbols.*

Content

Topic 17: Programming

17.1 Fundamental concepts	
Content	**Learners should:**
17.1.1 Select and use appropriate operators and flow control constructs	Be able to: • select and use appropriate operators and flow control constructs.
17.1.2 Select and use appropriate data types, variables and constants	Be able to: • select and use appropriate data types, variables and constants.
17.1.3 Select and use appropriate one- and two-dimensional data structures	Be able to: • select and use appropriate one- and two- dimensional data structures.

17.2 Creating solutions	
Content	**Learners should:**
17.2.1 Using computational thinking techniques	Be able to: • select and use appropriate computational thinking techniques.
17.2.2 Using IDEs and other tools	Be able to: • select and use appropriate tools.
17.2.3 Using libraries, files, modules and subprograms	Be able to: • select and use appropriate libraries, files, modules and subprograms.

Content

Topic 18: Representing and handling data

18.1 Data handling methods	
Content	**Learners should:**
18.1.1 Methods to handle floating-point data: **a.** Decimal module **b.** Rounding **c.** Decimal class **d.** Data type conversions **e.** Exception handling **f.** Compare	Be able to: • select and use appropriate methods for handling floating-point data. Further details provided in the Programming Language Subset (PLS) document.
18.1.2 Methods to handle regular expressions (re): **a.** Finding **b.** Validating	Be able to: • select and use appropriate methods for handling regular expressions. Further details provided in the Programming Language Subset (PLS) document.
18.1.3 Methods to handle numerical arrays (NumPy): **a.** Array **b.** Arithmetic **c.** Rounding **d.** Shape **e.** Load and save text files	Be able to: • select and use appropriate methods for handling numerical arrays. Further details provided in the Programming Language Subset (PLS) document.
18.1.4 Methods to handle data analysis (Pandas): **a.** Load CSV **b.** Measures of central tendency **c.** Information about data **d.** Clean data **e.** Load and save text files	Be able to: • select and use appropriate methods for handling data to be analysed. Further details provided in the Programming Language Subset (PLS) document.

18.2 Methods to represent and handle ADTs

Content	Learners should:
18.2.1 Methods to represent and handle a circular queue, implemented as a fixed-length list: **a.** Create **b.** Enqueue **c.** Dequeue **d.** IsEmpty **e.** IsFull	Be able to: • select and use appropriate methods to handle a circular queue, implemented as a fixed-length list. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of the circular queue ADT is assessed in Unit 3. Their ability to amend and write a circular queue implemented as a fixed-length list is assessed in Unit 4.
18.2.2 Methods to represent and handle a linked list: **a.** Represent: • Classes • References **b.** Handle: • Insert • Remove • Update • Find • Traverse	Be able to: • select and use appropriate methods to represent a linked list • select and use appropriate methods to handle a linked list. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of the linked list ADT is assessed in Unit 3. Their ability to amend and write a linked list is assessed in Unit 4.
18.2.3 Methods to represent and handle a hash table: **a.** Represent: • Classes • References **b.** Handle: • Calculate hash value • Insert • Remove • Find • Traverse	Be able to: • select and use appropriate methods to represent a hash table • select and use appropriate methods to handle a hash table. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of the hash table ADT is assessed in Unit 3. Their ability to amend and write a hash table is assessed in Unit 4.

18.2 Methods to represent and handle ADTs *(continued)*	
Content	**Learners should:**
18.2.4 Hash algorithm	Be able to: • select and use an appropriate hash algorithm. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of the concept of hashing is assessed in Unit 3. Their ability to amend and write hash algorithms is assessed in Unit 4.
18.2.5 Methods to handle collisions in a hash table: **a.** Separate chaining **b.** Linear probing **c.** Rehashing	Be able to: • select and use appropriate methods to handle collisions in a hash table. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of the hash table ADT is assessed in Unit 3. Their ability to amend and write code to handle collisions in a hash table is assessed in Unit 4.
18.2.6 Methods to represent a graph: **a.** Adjacency matrix **b.** Adjacency list	Be able to: • select and use appropriate methods to represent a graph. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of the graph ADT is assessed in Unit 3. Their ability to amend and write code to represent a graph is assessed in Unit 4.

18.2 Methods to represent and handle ADTs *(continued)*	
Content	**Learners should:**
18.2.7 Methods to represent and handle a binary search tree: **a.** Represent: • Classes • References **b.** Handle: • Insert • Find • Traverse	Be able to: • select and use appropriate methods to represent a binary search tree • select and use appropriate methods to handle a binary search tree. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of the tree ADT is assessed in Unit 3. Their ability to amend and write binary search trees is assessed in Unit 4.
18.2.8 Recursive tree traversal algorithms: **a.** Pre-order traversal **b.** In-order traversal **c.** Post-order traversal	Be able to: • select and use appropriate recursive algorithms to traverse a tree. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of the tree ADT is assessed in Unit 3. Their ability to amend and write recursive tree traversals is assessed in Unit 4.

Content

Topic 19: Best practice

19.1 Program design	
Content	**Learners should:**
19.1.1 Methods to produce correct outcomes: a. Problem decomposition into component parts b. Clear and appropriate logic c. Variables, constants, data types and data structures appropriate to the problem d. Programming constructs appropriate to the problem	Be able to: • select and use appropriate methods to produce correct outcomes to complex problems. Further details provided in the Good Programming Practice Guide (GPPG).
19.1.2 Methods to implement good practice in program design: a. Modularity b. Separation of concerns c. Minimisation of side effects d. Scope isolation e. Fitness for purpose of input and output f. Efficient solutions in terms of execution time and memory usage	Be able to: • select and use appropriate methods to design effective code. Further details provided in the Good Programming Practice Guide (GPPG).
19.1.3 Methods to implement modularity in source code: a. Blocked code b. User-defined subprograms c. Built-in subprograms d. Libraries of subprograms e. Files f. Procedures g. Functions	Be able to: • select and use appropriate methods to produce modular code. Further details provided in the Good Programming Practice Guide (GPPG).

19.2 Clean code	
Content	**Learners should:**
19.2.1 Consistency: **a.** Use of white space **b.** Line breaks layout **c.** Naming conventions **d.** Fixed-line length	Be able to: • select and use appropriate methods to promote code consistency. Further details provided in the Good Programming Practice Guide (GPPG).
19.2.2 Readability: **a.** Meaningful identifiers **b.** Informative comments **c.** Named constants **d.** Subprograms	Be able to: • use appropriate methods to promote readability of program code. Further details provided in the Good Programming Practice Guide (GPPG).
19.2.3 Methods to control behaviour of program code: **a.** Boundary value analysis **b.** Exception handling **c.** Minimising nested conditionals **d.** Minimising loops **e.** Using guard clauses **f.** Managed conversion of data types	Be able to: • select and use appropriate methods to control behaviour of code. Further details provided in the Programming Language Subset (PLS) document and Good Programming Practice Guide (GPPG).

19.3 Functionality	
Content	**Learners should:**
19.3.1 Methods to ensure that code is fit for purpose, produces accurate results and meets requirements: **a.** Testing and debugging **b.** Reviewing and refining	Be able to: • identify errors in program code • correct errors in program code • produce accurate outcomes • refine solutions to meet requirements. Further details provided in the Good Programming Practice Guide (GPG).

Content

Topic 20: Additional programming paradigms

20.1 Object-oriented programming (OOP) and functional programming	
Content	**Learners should:**
20.1.1 Methods for object-oriented programming (OOP): **a.** Classes **b.** Attributes **c.** Methods **d.** Objects (instance) **e.** Encapsulation **f.** Getters **g.** Setters **h.** Aggregation **i.** Composition **j.** Inheritance **k.** Polymorphism	Be able to: • convert Unified Modelling Language (UML) class diagrams into code • select appropriate methods for object-oriented programming • use appropriate methods for object-oriented programming. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of the characteristics of OOP is assessed in Unit 3. Their ability to amend and write OOP code in Python is assessed in Unit 4.
20.1.2 Methods for functional programming: **a.** Pure functions **b.** First-class functions **c.** Higher-order functions **d.** Immutable variables **e.** Avoid side effects **f.** Lazy evaluation **g.** Lambda functions **h.** Recursion **i.** Map **j.** Filter **k.** Reduce **l.** List comprehension **m.** Dictionary comprehension **n.** Zip	Be able to: • select appropriate methods for functional programming. • use appropriate methods for functional programming. Further details provided in the Programming Language Subset (PLS) document. Learners' understanding of the characteristics of functional programming is assessed in Unit 3. Their ability to amend and write functional code in Python is assessed in Unit 4.

Content

Topic 21: Algorithms

21.1 Sorting algorithms

Content	Learners should:
21.1.1 Recursive merge sort algorithm	Be able to: • implement and use a recursive merge sort algorithm. Learners' understanding of how the merge sort algorithm works is assessed in Unit 3. Their ability to implement a recursive merge sort in code is assessed in Unit 4.
21.1.2 Recursive quick sort algorithm	Be able to: • implement and use a recursive quick sort algorithm. Learners' understanding of how the quick sort algorithm works is assessed in Unit 3. Their ability to implement a recursive quick sort algorithm in code is assessed in Unit 4.

21.2 Shortest path and compression

Content	Learners should:
21.2.1 Dijkstra's shortest path algorithm	Be able to: • implement and use Dijkstra's shortest path algorithm. Learners' understanding of how the Dijkstra's shortest path algorithm works is assessed in Unit 3. Their ability to implement the algorithm in code is assessed in Unit 4.
21.2.2 Run-length encoding compression algorithm	Be able to: • implement and use a run-length encoding algorithm.

3 Assessment information

Assessment availability

This qualification offers exams as follows:

	International Advanced Subsidiary		International Advanced Level	
	Unit 1	Unit 2	Unit 3	Unit 4
June 2027	✓	✓	✘	✘
January 2028	✓	✓	✘	✘
June 2028	✓	✓	✓	✓
January 2029 onwards	✓	✓	✓	✓
June 2029 onwards	✓	✓	✓	✓

From June 2028, all four units will be assessed in January and June for the lifetime of the qualification.

Flexible assessment options

Learners taking this qualification have the option to undertake assessments in the following formats:

	International Advanced Subsidiary		International Advanced Level	
	Unit 1	Unit 2	Unit 3	Unit 4
Paper-based assessment	✓	✘	✓	✘
Onscreen assessment*	✓	✘	✓	✘
Practical programming **	✘	✓	✘	✓

* Onscreen assessment is available for this qualification. Find out more information on our website: qualifications.pearson.com.

** Refer to *the Pearson Instructions for Conducting Onscreen Examinations (ICOE)* document found on the *Pearson Edexcel International Advanced Level in Computer Science* website.

Unit 2 and Unit 4 practical programming

Programming environments

Learners taking Unit 2 and Unit 4 will complete onscreen practical programming assessments using Python 3 within an integrated development environment (IDE) of their choice. The use of Python has been selected for its wide adoption in education and industry, and for its suitability in developing core problem-solving and programming skills.

Centres are encouraged to choose an up-to-date version of Python 3 that is supported within a suitable IDE. Learners will require experience using an IDE to write, test and refine code during practical programming assessments. As these units are assessed onscreen, familiarity with the chosen IDE is essential for supporting productivity and problem-solving.

Centres must ensure that all AI-enabled tools, including locally installed large language models (LLM) or other automated code-generation features, are fully disabled during examination conditions. Our assessments are designed to be secure, and it is essential that they are conducted under controlled conditions. Failure to meet these conditions include the use of unauthorised tools that results in malpractice and/or maladministration. For further information about required examination conditions, please refer to the *Pearson Instructions for Conducting Onscreen Examinations (ICOE)* document found on the *Pearson Edexcel International Advanced Level in Computer Science* website.

A suitable IDE should include the following core functionalities:

- an editor with line numbering
- syntax highlighting and error checking
- tools for setting breakpoints and stepping through code
- a variable inspector or equivalent.

There is a wide range of freely available IDEs suitable for Python development. Teachers are encouraged to evaluate a selection and choose one that best suits the needs and abilities of their learners.

Learners should be confident in navigating and using their selected IDE independently before taking the onscreen practical assessments.

Programming Language Subset (PLS)

The Programming Language Subset (PLS) is a document that defines the specific elements of Python 3 required for this qualification. It outlines a core set of programming constructs that are sufficient to successfully complete all programming tasks in Unit 2 and Unit 4. The scope of the PLS is not intended to be extended beyond what is specified.

Teachers and learners are strongly encouraged to familiarise themselves with the PLS document. All questions have been designed to be solvable using the constructs specified in the PLS which reflect a common foundation found in most high-level programming languages. Learners may use constructs beyond those listed in the PLS. Valid solutions that incorporate additional or more advanced features will still be fully credited, provided they meet the requirements of the question.

The PLS is intended to provide clarity and focus for both teaching and assessment, supporting the consistent development of learners' programming skills. It will remain valid for the lifetime of the qualification. If updates are required, the latest version will be published on the *Pearson Edexcel International Advanced Level in Computer Science* qualification page on our website:

- in January for the June examination series
- in May for the January examination series.

Good Programming Practice Guide (GPPG)

The Good Programming Practice Guide (GPPG) is a document that complements the PLS by offering more detailed guidance on the core programming constructs. It provides more in-depth information and a wide range of examples. Teachers are encouraged to adapt and modify these examples to meet the specific needs of their learners, thereby enhancing teaching and learning experiences.

During the practical programming examinations, learners will only be provided with a digital version of the PLS. The GPPG must not be used during the examination.

The PLS and GPPG are available to download in the *Pearson Edexcel International Advanced Level in Computer Science* qualification page on our website.

Pearson Instructions for Conducting Onscreen Examinations (ICOE)

For instructions on setting up learners' secure user areas, downloading the coding and PLS files, administering the examination, and submitting completed files to Pearson for marking, please see the *Pearson Instructions for Conducting Onscreen Examinations (ICOE)* document on the *Pearson Edexcel International Advanced Level in Computer Science* qualification page on our website.

Security and backups

It is the centre's responsibility to keep the learners' work that has been submitted for assessment secure.

For materials stored electronically, centres are strongly advised to use firewall protection and virus-checking software, and to employ an effective backup strategy so that an up-to-date archive of learners' evidence is maintained.

Assessment objectives

The examinations will measure how learners have achieved the following assessment objectives.

Assessment Objective (AO)	
AO1	Demonstrate knowledge and understanding of the key principles and concepts of computer science.
AO2	Apply knowledge and understanding of key principles and concepts of computer science, including to solve programming problems.
AO3	Design, program, test and refine programmed solutions using different types of programming paradigms.

Assessment objectives for International Advanced Subsidiary (IAS) and Advanced Level (IAL)

Assessment Objective	Weighting in IAS %	Weighting in IAL %
AO1	27.5	27.5
AO2	42.5	40
AO3	30	32.5
Total	100	100

Breakdown of assessment objectives

	Weighting in each unit %			
Assessment Objective	Unit 1	Unit 2	Unit 3	Unit 4
AO1	55	0	55	0
AO2	45	40	45	30
AO3	0	60	0	70
Total	100	100	100	100

4 Administration and general information

Entries

Details of how to enter learners for the examinations for this qualification can be found in our *International Information Manual*. A copy is made available to all examinations officers and is available on our website: qualifications.pearson.com.

Resitting of units

Learners can resit any unit irrespective of whether the qualification is to be cashed in. If a learner resits a unit more than once, only the better of the two most recent attempts of that unit will be available for aggregation to a qualification grade.

Access arrangements, reasonable adjustments, special consideration and malpractice

Equality and fairness are central to our work. Our policies require all learners to have equal opportunity to access our qualifications and assessments, and our qualifications to be awarded in a way that is fair to every learner.

We are committed to ensuring that learners with a protected characteristic, as defined by the Equality Act 2010, are not disadvantaged when undertaking one of our qualifications, and that all learners receive the recognition they deserve for their achievement, which can be compared fairly with that of their peers.

Language of assessment

Assessment of this qualification will be available in English. All learners' work must be in English.

We recommend that learners are able to read and write in English at Level B2 of the Common European Framework of Reference for Languages.

Access arrangements

Access arrangements are agreed before an assessment. They allow learners with special educational needs, disabilities or temporary injuries to:

- access the assessment
- show what they know and can do without changing the demands of the assessment.

The intention behind an access arrangement is to meet the particular needs of an individual learner with a disability, without affecting the integrity of the assessment. Access arrangements are the principal way in which awarding bodies comply with the duty under the Equality Act 2010 to make 'reasonable adjustments'.

Access arrangements should always be processed at the start of the course. Learners will then know what is available and have the access arrangement(s) in place for assessment.

Reasonable adjustments

The Equality Act 2010 requires an awarding organisation to make reasonable adjustments where a person with a disability would be at a substantial disadvantage in undertaking an assessment. The awarding organisation is required to take reasonable steps to overcome that disadvantage.

A reasonable adjustment for a particular person may be unique to that individual and therefore might not be in the list of available access arrangements.

Whether an adjustment will be considered reasonable will depend on a number of factors, including:

- the needs of the learner with the disability
- the effectiveness of the adjustment
- the cost of the adjustment and
- the likely impact of the adjustment on the learner with the disability and other learners.

An adjustment will not be approved if it involves unreasonable costs to the awarding organisation or affects timeframes or the security or integrity of the assessment. This is because the adjustment is not 'reasonable'.

Special consideration

Special consideration is a post-examination adjustment to a learner's mark or grade to reflect temporary injury, illness or other indisposition at the time of the examination/ assessment, which has had, or is reasonably likely to have had, a material effect on a learner's ability to take an assessment or demonstrate their level of attainment in an assessment.

Further information

Information about how to apply for access arrangements and special consideration, please refer to our website: https://qualifications.pearson.com/en/support/support-topics/understanding-our-qualifications/policies-for-centres-learners-and-employees.html.

For further information about access arrangements, reasonable adjustments and special consideration, please refer to the JCQ website: www.jcq.org.uk.

Candidate malpractice

Candidate malpractice refers to any act by a candidate that compromises or seeks to compromise the process of assessment, or which undermines the integrity of the qualifications or the validity of results/certificates.

Candidate malpractice in examinations **must** be reported to Pearson using a *JCQ Form M1* (available at www.jcq.org.uk/exams-office/malpractice). The form should be emailed to candidatemalpractice@pearson.com. Please provide as much information and supporting documentation as possible. Note that the final decision regarding appropriate sanctions lies with Pearson.

Failure to report malpractice constitutes staff or centre malpractice.

Staff/centre malpractice

Staff and centre malpractice includes both deliberate malpractice and maladministration of our qualifications. As with candidate malpractice, staff and centre malpractice is any act that compromises or seeks to compromise the process of assessment, or which undermines the integrity of the qualifications or the validity of results/certificates.

All cases of suspected staff malpractice and maladministration **must** be reported immediately, before any investigation is undertaken by the centre, to Pearson on a *JCQ Form M2(a)* (available at www.jcq.org.uk/exams-office/malpractice). The form, supporting documentation and as much information as possible should be emailed to pqsmalpractice@pearson.com. Note that the final decision regarding appropriate sanctions lies with Pearson.

Failure to report malpractice itself constitutes malpractice.

More detailed guidance on malpractice can be found in the latest version of the document *General and Vocational Qualifications Suspected Malpractice in Examinations and Assessments Policies and Procedures*, available at http://www.jcq.org.uk/exams-office/malpractice.

Awarding and reporting

Pearson Edexcel International Advanced Subsidiary in Computer Science will be graded on a five-grade scale from A to E. The Pearson Edexcel International Advanced Level in Computer Science will be graded on a six-point scale A* to E. Individual unit results will be reported. Only Units 1 and 2 will contribute to the International Advanced Subsidiary grade. All four units, 1, 2, 3 and 4 will contribute to the International Advanced Level grade.

The first certification opportunity for Pearson Edexcel International Advanced Subsidiary in Computer Science will be in August 2027. The first certification opportunity for the Pearson Edexcel International Advanced Level in Computer Science will be in August 2028. A pass in an International Advanced Subsidiary subject is indicated by one of the five grades A, B, C, D, E, of which grade A is the highest and grade E the lowest. A pass in an International Advanced Level subject is indicated by one of the six grades A*, A, B, C, D, E, of which grade A* is the highest and grade E the lowest. Learners whose level of achievement is below the minimum judged by Pearson to be of sufficient standard to be recorded on a certificate will receive an unclassified U result.

Unit results

Learners will receive a uniform mark between 0 and the maximum uniform mark for each unit. The uniform marks at each grade threshold for each unit are:

Unit grade	Maximum uniform mark	A	B	C	D	E
	100	80	70	60	50	40

Qualification results

The minimum uniform marks required for each grade:
International Advanced Subsidiary (cash-in code: XCP01)

Qualification grade	Maximum uniform mark	A	B	C	D	E
	200	160	140	120	100	80

Learners with a uniform mark in the range 0–79 will be Unclassified (U).
International Advanced Level (cash-in code: YCP01)

Qualification grade	Maximum uniform mark	A	B	C	D	E
	400	320	280	240	200	160

Learners with a uniform mark in the range 0–159 will be Unclassified (U).

To be awarded an A*, learners will need to achieve an A for the International Advanced Level qualification (at least 320 uniform marks) and at least 90 per cent of the total uniform mark available across the IA2 units combined (at least 180 uniform marks).

Learner recruitment and progression

Pearson follows the Joint Council for Qualifications (JCQ) policy concerning recruitment to our qualifications in that:

- they must be available to anyone who is capable of reaching the required standard
- they must be free from barriers that restrict access and progression
- equal opportunities exist for all learners.

Prior learning and other requirements

There are no prior learning or other requirements for this qualification.

Learners who would benefit most from studying this qualification are likely to have a Level 2 qualification, such as a GCSE in Computer Science or equivalent.

Progression

Learners can progress from these qualifications to:

- further studies in Computer Science as well as to a wide range of other subjects
- employment
- further training

Appendices

Appendix 1:

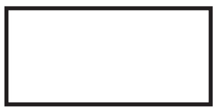
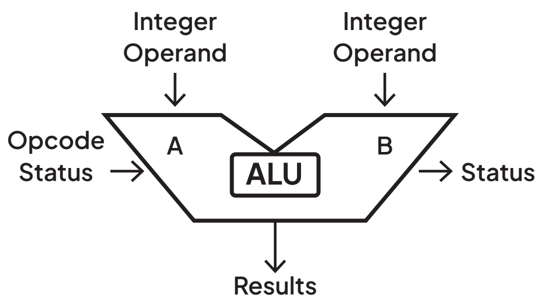
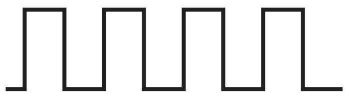
Codes

Type of code	Use of code	Unit	Code
Unit codes	Each unit is assigned a unit code. This unit code is used as an entry code to indicate that a learner wishes to take the assessment for that unit. Centres will need to use the entry codes only when entering learners for their examination.	Unit 1	Written: WCP01/01
			Onscreen: WCP01/01C
		Unit 2	WCP02/01
		Unit 3	Written: WCP03/01
			Onscreen: WCP03/01C
		Unit 4	WCP04/01
Cash-in codes	The cash-in code is used as an entry code to aggregate the learner's unit scores to obtain the overall grade for the qualification. Centres will need to use the entry codes only when entering learners for their qualification.	International Advanced Subsidiary	XCP01
		International Advanced Level	YCP01
Entry codes	The entry codes are used to: • enter a learner for the assessment of a unit • aggregate the learner's unit scores to obtain the overall grade for the qualification.	Please refer to the Pearson Edexcel Information Manual, available on the Pearson qualifications website.	

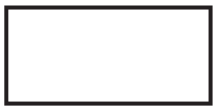
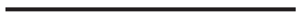
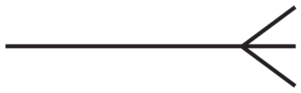
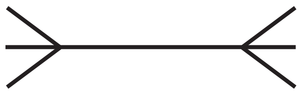
Appendix 2: Symbols

Appendix 2 shows the symbols that will be used in each unit.

Unit 1

Symbols for CPU components	
Purpose	**Symbol**
Register	
ALU	 Integer Operand Integer Operand Opcode Status A ALU B Status Results
Clock signal	

Unit 1 and Unit 2

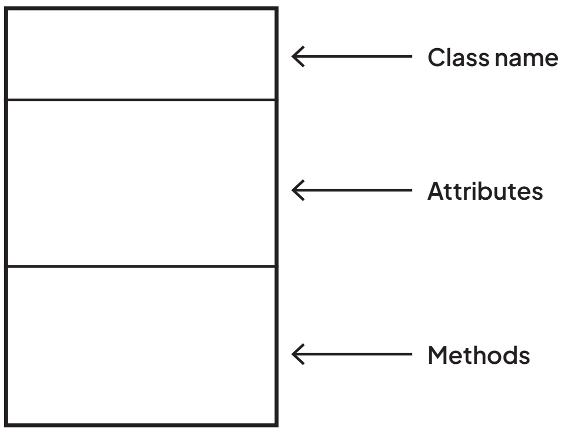
Entity-relationship-diagram (ERD)	
Purpose	**Symbol**
Entity	
1-to-1 relationship (1:1)	
1-to-many relationship (1:m)	
Many-to-many relationship (m:m)	

Unit 1 and Unit 3

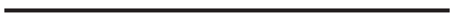
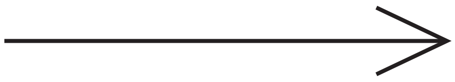
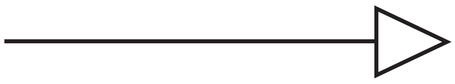
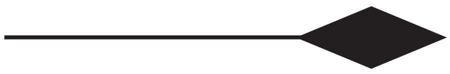
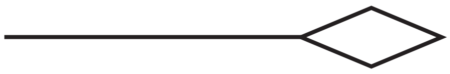
Register transfer notation
[ ] The content of the component inside the brackets is loaded into the component on the left pointed at by the arrow e.g. MAR <– [PC] Contents of the program counter (PC) register is loaded into the memory address register (MAR) [CIR] Contents of CIT decoded and executed.

Boolean algebra notation		
Operator		**Notation**
AND	A **AND** B	$A \,.\, B$
OR	A **OR** B	$A + B$
NOT	**NOT** A	$\overline{A}$
NAND	A **NAND** B	$\overline{A \,.\, B}$
NOR	A **NOR** B	$\overline{A + B}$
XOR	A **XOR** B	$A \oplus B$

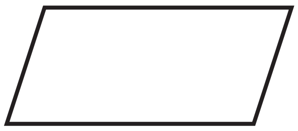
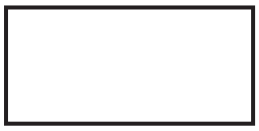
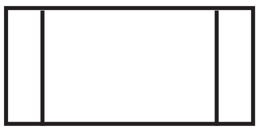
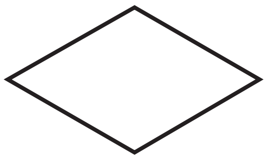
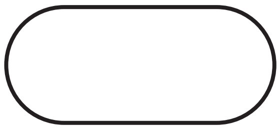
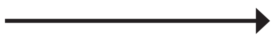
Unit 3 and Unit 4

UML class diagram


Visibility notation (access modifiers)
• private: - • protected: # • public: +

Relationships	
Association	
Directed association	
Inheritance/ generalisation	
Composition	
Aggregation	

Units 1, 2, 3 and 4

Flowchart	
Purpose	**Symbol**
Input/Output	
Process	
Subprocess	
Decision	
Terminator	
Labels	**Yes** **No**
Flow	

Appendix 3:

Assembly language instruction set

In the interest of giving learners real-life experiences, we have opted to reference ARMlite, which is a cut-down version of the assembly language instruction set used for programming the ARM chips that are found in a multitude of different devices, including Raspberry Pi computers. Teachers may choose to teach this in a purely theoretical way.

This list of Assembler instructions is a subset of the instruction set for the ARMlite simulator written by Peter Higginson.

ARMlite simulates a simple computer built around a cut-down version of a 32–bit Arm processor. It has 13 general-purpose registers (R0 – R12) and three special-purpose registers: The first operand for all data processing operations must be a register. The second operand can be either an immediate value (#x) or a register (Rn).

Immediate values may be specified in decimal, hex or binary format, preceded by a #.

Comments are prefixed by a semicolon ;. They may be added after an instruction or on a line on their own.

Labels are used to specify the location of instructions for branching and to specify locations for loading and storing data. Labels must start with a letter and may be made up of upper- and lower-case letters and numbers.

Arithmetic and logical operations	
ADD Rd, Rn, <operand>	Add the value in *<operand>* to the value in register Rn and store the result in register Rd.
SUB Rd, Rn, <operand>	Subtract the value in *<operand>* from the value in register Rn and store the result in register Rd.
AND Rd, Rn, <operand>	Perform a bitwise logical AND between the values in register Rn and *<operand>*, and store the result in register Rd.
ORR Rd, Rn, <operand>	Perform a bitwise logical OR between the values in register Rn and *<operand>*, and store the result in register Rd.
EOR Rd, Rn, <operand>	Perform a bitwise logical XOR between the values in register Rn and *<operand>*, and store the result in register Rd.
MVN Rd, <operand>	Performs a bitwise logical NOT to the value in *<operand>*, and store the result in register Rd.
LSL Rd, Rn, <operand>	Logically shift the value stored in register Rn to the left by the number of bits specified by *<operand>* and store the result in register Rd.
LSR Rd, Rn, <operand>	Logically shift the value stored in register Rn to the right by the number of bits specified by *<operand>* and store the result in register Rd.
Comparison and branch instructions	
CMP Rn, <operand>	Compare the value in *<operand>* with the value stored in register Rn and set status bits based on the outcome.
B *<label>*	Unconditionally branch to the address at position *<label>*.
B*<condition>* *<label>*	Conditionally branch to the instruction at position *<label>* when the last CMP instruction result matches the specified *<condition>*.
Data movement and other instructions	
MOV Rd, *<operand>*	Move the value in *<operand>* into register Rd.
LDR Rd, *<address>*	Load the contents of the memory location *<address>* into register Rd.
STR Rd, *<address>*	Store the contents of Rd to the memory location *<address>*.
HALT	Halt program execution.
Syntax notes	
<operand>	References either *#x* decimal value or *Rn* register n
<condition>	EQ equal to, NE not equal to, GT greater than, LT less than
<label>:	A label definition is formed by an identifier followed by a colon
Rd, Rn	General purpose registers R0 to R12

Appendix 4: Glossary

Assessment glossary

Term	Definition
Assessment objectives	The requirements that learners need to meet to succeed in the qualification. Each assessment objective has a unique focus, which is then targeted in examinations or non-examined assessment (NEA). Assessment objectives may be assessed individually or in combination. Assessment Objectives refer to the skills and knowledge that learners need to demonstrate in their responses to questions in the assessments.
Cash-in codes	Cash-in codes are used in combination with entry codes to aggregate the learner's unit scores to obtain the overall grade for the qualification.
Entry codes	To enter the learner for their examination, unit codes are used as entry codes. To obtain the overall grade for the qualification, entry codes are used in combination with cash-in codes.
International Advanced Subsidiary	Abbreviated to IAS.
International Advanced Level	Abbreviated to IAL.
International A2 (IA2)	Additional content required for an IAL.
Modular	Modular qualifications contain units of assessment. These units can be taken during the course of study. The final qualification grade is worked out from the combined unit results.
Unit code	Each unit is assigned a unit code. This unit code is used as an entry code to indicate that a learner wishes to take the assessment for a particular unit.
Uniform mark	A learner's actual marks (or raw marks) will be converted into UMS mark so that it is possible to see the proportionate result of a learner. The raw marks for each unit may differ, but the uniform mark will be the same.
Unit	A modular qualification will be divided into a number of units. Each unit will have its own assessment.

Subject content glossary

Term	Definition
Benefits	Positive outcomes or effects gained from using the specified item. The advantages that it confers.
Characteristics	The properties, features, attributes or traits of the specified item. What it is like or how it behaves.
Definition	The meaning of the specified item. What it is.
Drawbacks	Negative outcomes or effects resulting from the use of the specified item. The disadvantages that it results in.
Function	Specific action(s) or task(s) performed by the specified item. What it does.
Item	A component, concept, device, method, term or a technique.
Operation	The sequence of steps or processes carried out by the specified item. How it performs its function.
Why needed	The reason, rationale or necessity for the specified item. Why it is needed.

Appendix 5: Command word taxonomy

The following command words in this taxonomy will be used consistently by Pearson in its assessments to ensure learners are rewarded for demonstrating the necessary skills. Careful consideration has been given to this taxonomy to ensure that Assessment Objectives are targeted consistently across questions.

Please note: the list below will not necessarily be used in every unit/session and is provided for guidance only.

Unit 1 and 3

Command word	Definition/meaning
Complete	Requires the completion of an artefact.
Calculate	Obtain a numerical answer, showing relevant working. If the answer has a unit, this must be included
Construct	Requires creation of an expression using subject specific symbolic representation, rules and syntax.
Convert	Requires changing information from one symbolic representation to another representation.
Define	Requires giving the meaning of a term from the specification.
Outline	To give an account of something. Statements in the response need to be linked but do not need to include a justification or reason.
Describe	To give an account of a process. Statements in the response need to be in order and linked, but do not need to include a justification or reason.
Explain	An explanation requires a justification of a point. The answer must contain some element of reasoning/justification, this can include mathematical/logical explanations.
Give/State	All these command words are really synonyms. They all require recall of one or more pieces of information.
Identify	Usually requires some key information to be selected from a given stimulus/resource/set of options.
Simplify	Requires application of logic to reduce an expression.
Which	Requires selection of a response (or responses) from a given set of options.

Unit 2 and 4

Command word	Definition/meaning
Amend	Requires changes or additions to code, or deletions or rearrangement of code.
Write	Requires creation/manipulation of code using a high-level programming language.

Appendix 6:

Transferable skills

The need for transferable skills

In recent years, higher-education institutions and employers have consistently flagged the need for learners to develop a range of transferable skills to enable them to respond with confidence to the demands of undergraduate study and the world of work. The Organisation for Economic Co-operation and Development (OECD) defines skills, or competencies, as 'the bundle of knowledge, attributes and capacities that can be learned and that enable individuals to successfully and consistently perform an activity or task and can be built upon and extended through learning.'[1] To support the design of our qualifications, the Pearson Research Team selected and evaluated seven global 21st-century skills frameworks. Following on from this process, we identified the National Research Council's (NRC) framework [2] as the most evidence-based and robust skills framework, and have used this as a basis for our adapted skills framework. The framework includes cognitive, intrapersonal skills and interpersonal skills.

COGNITIVE
INTRAPERSONAL
INTERPERSONAL

The skills have been interpreted for this specification to ensure they are appropriate for the subject. All of the skills listed are evident or accessible in the teaching, learning and/or assessment of the qualification. Some skills are directly assessed. Pearson materials will support you in identifying these skills and developing these skills in learners.

The table on the next page sets out the framework and gives an indication of the skills that can be found in chemistry and indicates the interpretation of the skill in this area. A full subject interpretation of each skill, with mapping to show opportunities for learners' development is provided on the subject pages of our website: qualifications.pearson.com

[1] OECD – *Better Skills, Better Jobs, Better Lives: A Strategic Approach to Skills Policies* (OECD Publishing, 2012)

[2] Koenig, J. A. (2011) – *Assessing 21st Century Skills: Summary of a Workshop* (National Academies Press, 2011)

Cognitive skills	Cognitive processes and strategies	• Critical thinking • Problem solving • Analysis • Reasoning • Interpretation • Decision making • Adaptive learning • Executive function
	Creativity	• Creativity • Innovation
Intrapersonal skills	Intellectual openness	• Adaptability • Personal and social responsibility • Continuous learning • Intellectual interest and curiosity
	Work ethic/ conscientiousness	• Initiative • Self-direction • Responsibility • Perseverance • Productivity • Self-regulation (metacognition, forethought, reflection) • Ethics • Integrity
	Positive core self-evaluation	• Self-monitoring/ selfevaluation/ selfreinforcement
Interpersonal skills	Teamwork and collaboration	• Communication • Collaboration • Teamwork • Cooperation • Interpersonal skills
	Leadership	• Leadership • Responsibility • Assertive communication • Self-presentation

Problem solving in the application of unifying patterns and themes in chemistry and using them in new and changing situations.

Initiative when using knowledge of chemistry, independently (without guided learning), to further own understanding.

Communication to convey a science process or technique (verbally or in writing) to peers and teachers and answer questions from others.

About Pearson

We are the world's leading learning company operating in countries all around the world. We provide content, assessment and digital services to learners, educational institutions, employers, governments and other partners globally. We are committed to helping equip learners with the skills they need to enhance their employability prospects and to succeed in the changing world of work. We believe that wherever learning flourishes, so do people.

Publication code: GQ000050

Cover image: Shutterstock / archy13