Year 10 Overview 2025-26 – *GCSE Computer Science*				
Date	**Wk**	**Week**	**Units Studied & Learning Outcomes**	**Key Concepts & Assessment**
8 weeks (20 Lessons) (38Days)				
Tues 2-Sep Y7 only Wed-whole school	A	1	• Overview of Unit/No. lessons Topic 1: Computational thinking Topic 6: Problem solving with programming Topic 2: Data • Lesson **Sequence** of Content: Lesson 1:2 - Intro to programming Lesson 3:4 - Decomposition, algorithms Lesson 5:6 - Data types, variables Lesson 6:7 - Input and integer functions Lesson 8:9 - Flowcharts Lesson 10:11 - Binary Lesson 12:13 - Unsigned integers Lesson 14:15 - Binary arithmetic Lesson 16:17 - Two's complement Lesson 18:19 – End of unit assessment • Unit Learning Outcomes: ➢ understand and apply the fundamental principles and concepts of computer science, including abstraction, decomposition, logic, algorithms, and data representation ➢ analyse problems in computational terms through practical experience of solving such problems, including designing, writing and debugging programs.	• Foundational Concepts Topic 1 & 6: Students are expected to develop a set of computational thinking skills that enable them to design, implement and analyse algorithms for solving problems. Students should be competent at designing, reading, writing and debugging programs. They must be able to apply their skills to solve real problems and produce readable, robust programs. Topic 2: Computers use binary to represent different types of data. Students are expected to learn how different types of data are represented in a computer. • Key vocabulary program, programming language, integrated development environment, IDE, run, change, save, arithmetic operator, BIDMAS, readable code, maintainable code, variable, decomposition, algorithm, sequence, error message, debug, primitive data type, integer, real, char, string, identifier, assignment, input, output, runtime error, flowchart, symbol, binary, digital computer, bit, byte, nibble, denary, binary conversion, binary arithmetic, overflow error, two's complement, signed integer, unsigned integer, positive binary, negative binary • Commentary ✓ Access an integrated development environment ✓ Load, run, change, and save a Python program ✓ Use arithmetic operators and BIDMAS ✓ Layout code to be readable and maintainable ✓ Correct errors in programs and interpret error messages ✓ Use variables in algorithms and programs ✓ Define decomposition and algorithm ✓ Decompose a problem and sequence algorithm steps (unplugged + IDE) ✓ Recognise and create variables of primitive data types (int, real, char, string) ✓ Use meaningful identifiers and assign correct data types ✓ Take input and produce output ✓ Understand and fix runtime errors ✓ Translate between flowcharts and code ✓ Represent algorithms visually in flowcharts ✓ Define binary, bit, nibble, byte ✓ Convert between binary and denary (8-bit) ✓ Explain use of binary and calculate 2^n ✓ Add binary numbers and identify overflow errors ✓ Understand signed vs unsigned integers ✓ Represent numbers in two's complement and find two's complement of positives • **Assessment –** Informal assessment Sample assessment materials will be available to help prepare learners for the formal assessment
15-Sep (INSET Friday)	A	3		
22-Sep	B	4		
29-Sep	A	5		
6-Oct	B	6		
13-Oct	A	7		
20-Oct	B	8		
8-Sep	B	2		

<table>
<tr><th>Half-Term</th><th colspan="3">7 weeks (17-18 lessons) (35 Days)</th></tr>
<tr><td>3-Nov</td><td>A</td><td>9</td><td rowspan="7">• Overview of Unit/No. lessons

Topic 6: Problem solving with programming

Topic 2: Data

• Lesson Sequence of Content:
Lesson 1:2 - String methods
Lesson 3:4 - if, if else, operators
Lesson 5:6 - if elif else, readability
Lesson 6:7 - Repetition (while)

Lesson 8:9 - Two’s complement
Lesson 10:11 – Binary shifts (logical / arithmetic)
Lesson 12:13 - Hexadecimal
Lesson 14:15 - ASCII

Lesson 16:17 – End of unit assessment

• Unit Learning Outcomes:
➢ understand and apply the fundamental principles and concepts of computer science, including abstraction, decomposition, logic, algorithms, and data representation
➢ analyse problems in computational terms through practical experience of solving such problems, including designing, writing and debugging programs.</td><td rowspan="7">• Foundational Concepts
Topic 6: Students should be competent at designing, reading, writing and debugging programs. They must be able to apply their skills to solve real problems and produce readable, robust programs.

Topic 2: Computers use binary to represent different types of data. Students are expected to learn how different types of data are represented in a computer.

• Key vocabulary
string manipulation, string methods, index, slicing, upper, lower, isalpha, relational operators, if, if else, elif, decision symbol, Boolean operators, AND, OR, NOT, truth table, condition-controlled loop, while loop, two’s complement, signed integer, binary conversion, range, logical shift, arithmetic shift, left shift, right shift, hexadecimal, base 16, binary-hex conversion, ASCII, character set, 7-bit encoding, Unicode

• Commentary
✓ Use string manipulation methods (indexing, case conversion, validation)
✓ Use relational operators in flowcharts and code
✓ Implement if, if else, and if elif else statements
✓ Identify and apply flowchart decision symbols
✓ Improve code readability with whitespace, comments, indentation, identifiers
✓ Recognise and label elements of code (variables, constants, selection, repetition)
✓ Define and apply Boolean logic (AND, OR, NOT) and construct truth tables
✓ Use condition-controlled loops (while) in flowcharts, algorithms, and code
✓ Convert between signed denary and two’s complement binary
✓ Calculate ranges for two’s complement values
✓ Perform logical shifts to multiply/divide unsigned binary numbers
✓ Explain precision loss from logical right shifts
✓ Perform arithmetic shifts on signed binary numbers
✓ Explain difference between logical and arithmetic right shifts
✓ Define hexadecimal and convert between binary and hex
✓ Explain the purpose and use of hexadecimal notation
✓ Define character sets and how characters are represented in ASCII
✓ Identify ASCII values for characters and explain limitations of 7-bit ASCII

• Assessment – Informal assessment
Sample assessment materials will be available to help prepare learners for the formal assessment</td></tr>
<tr><td>10-Nov</td><td>B</td><td>10</td></tr>
<tr><td>17-Nov</td><td>A</td><td>11</td></tr>
<tr><td>24-Nov</td><td>B</td><td>12</td></tr>
<tr><td>1-Dec</td><td>A</td><td>13</td></tr>
<tr><td>8-Dec</td><td>B</td><td>14</td></tr>
<tr><td>15-Dec</td><td>A</td><td>15</td></tr>
</table>

<table>
<tr><td colspan="2">Christmas Holiday</td><td colspan="2">6 weeks (15 lessons) (30 Days)</td><td></td></tr>
<tr><td>5-Jan</td><td>B</td><td>16</td><td rowspan="6">• <u>Overview of Unit/No. lessons</u>
Topic 6: Problem solving with programming
Topic 3: Computers

• <u>Lesson Sequence of Content</u>:
Lesson 1:2 - One-dimensional lists
Lesson 3:4 - for loops
Lesson 5:6 - Procedures / Functions
Lesson 6:7 - Procedures / Functions
Lesson 8:9 - Subprograms

Lesson 10:11 - Fetch-decode-execute
Lesson 12:13 - Secondary storage

Lesson 14:15 – End of unit assessment

• <u>Unit Learning Outcomes</u>:
➢ understand and apply the fundamental principles and concepts of computer science, including abstraction, decomposition, logic, algorithms, and data representation
➢ analyse problems in computational terms through practical experience of solving such problems, including designing, writing and debugging programs.
➢ understand the components that make up digital systems and how they communicate with one another and with other systems</td><td rowspan="6">• Foundational Concepts
Topic 6: Learning to program is a core component of a computer science course. Students should be competent at designing, reading, writing and debugging programs. They must be able to apply their skills to solve real problems and produce readable, robust programs.

Topic 3: Students must be familiar with the hardware and software components that make up a computer system.

• Key vocabulary
list, array, indexing, append, delete, range(), for loop, iteration, procedure, parameter, function, return value, subprogram, separation of concerns, stored program concept, von Neumann architecture, CPU, fetch-decode-execute cycle, clock speed, pipelining, address bus, memory address, addressable memory

• Commentary
✓ Define and use lists (arrays), access and modify elements using indexing
✓ Use range() to generate sequences, apply for loops for iteration
✓ Define and create procedures with and without parameters
✓ Define and create functions that return values
✓ Use subprograms to organise code and apply the concept of ‘separation of concerns’
✓ Explain the stored program concept and von Neumann architecture
✓ Identify hardware components and explain roles in the fetch-decode-execute cycle
✓ Explain how clock speed and pipelining affect CPU performance
✓ Understand the relationship between address bus width and addressable memory
✓ Calculate addressable memory locations based on address bus size

• Assessment – Informal assessment
Sample assessment materials will be available to help prepare learners for the formal assessment</td></tr>
<tr><td>12-Jan</td><td>A</td><td>17</td></tr>
<tr><td>19-Jan</td><td>B</td><td>18</td></tr>
<tr><td>26-Jan</td><td>A</td><td>19</td></tr>
<tr><td>2-Feb</td><td>B</td><td>20</td></tr>
<tr><td>9-Feb</td><td>A</td><td>21</td></tr>
<tr><td colspan="2">Half-Term</td><td colspan="2">6 weeks (15 lessons) (28 Days)</td><td></td></tr>
<tr><td>23-Feb</td><td>B</td><td>22</td><td rowspan="2">• <u>Overview of Unit/No. lessons</u>
Topic 6: Problem solving with programming
Topic 3: Computers

• <u>Lesson Sequence of Content</u>:
Lesson 1:2 - string.format()
Lesson 3:4 - Two-dimensional lists
Lesson 5:6 - Validation
Lesson 6:7 - Linear search (one-dimensional)</td><td rowspan="2">• Foundational Concepts
Topic 6: Learning to program is a core component of a computer science course. Students should be competent at designing, reading, writing and debugging programs. They must be able to apply their skills to solve real problems and produce readable, robust programs.

Topic 3: Students must be familiar with the hardware and software components that make up a computer system.

• Key vocabulary
operating system, peripheral, driver, user interface, access control, authentication, utility software, file management,</td></tr>
<tr><td>2-Mar</td><td>A</td><td>23</td></tr>
</table>

9-Mar	B	24	Lesson 8:9 - Linear search (two-dimensional) Lesson 10:11 - Operating systems Lesson 12:13 - Utility software Lesson 14:15 – End of unit assessment • Unit Learning Outcomes: ➢ understand and apply the fundamental principles and concepts of computer science, including abstraction, decomposition, logic, algorithms, and data representation ➢ analyse problems in computational terms through practical experience of solving such problems, including designing, writing and debugging programs. ➢ understand the components that make up digital systems and how they communicate with one another and with other systems	file permission, file access, process management, scheduling, round-robin, paging, file recovery, file compression, disk defragmentation, linear search, one-dimensional list, two-dimensional list, string formatting, indexing, validation, presence check, length check, range check, pattern check • Commentary ✓ Describe the role and tasks of an operating system ✓ Explain how the OS manages peripherals using drivers ✓ Understand file management and file permissions ✓ Explain process scheduling, round-robin, and paging in memory ✓ Define and identify utility software and match tools to specific tasks ✓ Apply linear search algorithms to one- and two-dimensional lists (paper, flowchart, and code) ✓ Use string.format() to display output in a structured and user-friendly way ✓ Define and use one- and two-dimensional data structures with indexing and iteration ✓ Validate user input using checks such as presence, length, range, and pattern • **Assessment –** x2 mock exam papers Papers will be generated to test the content covered to date, in Principles of Computer Science (written paper) & Application of Computational Thinking (coding paper)
16-Mar	A	25		
23-Mar	B	26		
30-Mar (finish Wed 1st April)	A	ST1		
Easter Holiday			5 weeks (12-13 lessons) (24 Days)	
20-Apr	B	ST1	• Overview of Unit/No. lessons Topic 1: Computational thinking Topic 6: Problem solving with programming Topic 5: Issues and impact • Lesson **Sequence** of Content: Lesson 1:2 - Merge sort Lesson 3:4 - Reading / Writing files Lesson 5:6 - Authentication Lesson 7:8 - Malware & anti-malware / Hackers / Social engineering Lesson 9:10 - Data-level protection / Robust software Lesson 11:12 – End of unit assessment • Unit Learning Outcomes: ➢ understand and apply the fundamental principles and concepts of computer science, including	• Foundational Concepts Topic 1 & 6: Students are expected to develop a set of computational thinking skills that enable them to design, implement and analyse algorithms for solving problems. Students should be competent at designing, reading, writing and debugging programs. They must be able to apply their skills to solve real problems and produce readable, robust programs. Topic 5: Students should be aware of the influence of digital technology and recognise some of the issues and the impact on wider society associated with its use. • Key vocabulary merge sort, read file, write file, comma-separated value (CSV), two-dimensional structure, authentication, record, cyberattack, malware, anti-malware, hacker, firewall, penetration testing, social engineering, phishing, pretexting, baiting, quid pro quo, acceptable use policy, encryption, backup, recovery, access control, robust software, vulnerability, audit trail • Commentary ✓ Describe and implement merge sort ✓ Read from and write to text files
27-Apr	A	29		
4-May (Bank holiday Mon)	B	30		

11-May	A	31	abstraction, decomposition, logic, algorithms, and data representation ➢ analyse problems in computational terms through practical experience of solving such problems, including designing, writing and debugging programs. ➢ understand the impact of digital technology on wider society, including issues of privacy and cybersecurity	✓ Process strings using split() and store them in two-dimensional lists ✓ Implement authentication using structured data and conditional logic ✓ Explain the nature and consequences of cyberattacks and malware ✓ Describe how anti-malware software and firewalls protect systems ✓ Identify social engineering tactics and explain the purpose of acceptable use policies ✓ Describe methods of data protection including encryption, backups, and access control ✓ Define robust software and recognise secure vs insecure coding practices • **Assessment –** Informal assessment Sample assessment materials will be available to help prepare learners for the formal assessment
18-May	B	32		
Half-Term			7 weeks (17-18 lessons) (35 Days)	
1-Jun	A	33	• Overview of Unit/No. lessons Topic 6: Problem solving with programming Topic 4: Networks • Lesson **Sequence** of Content: Lesson 1:2 - Turtle introduction, decomposing a big problem Lesson 3:4 - Turtle movement and coordinates Lesson 5:6 - Turtle pens and colours Lesson 6:7 - Turtle fill (begin_fill, end_fill) Lesson 8:9 - LANs and WANs Lesson 10:11 - Network speed Lesson 12:13 - Wired vs wireless Lesson 14:15 - Network topologies Lesson 16:17 – End of unit assessment • Unit Learning Outcomes: ➢ understand and apply the fundamental principles and concepts of computer science, including abstraction, decomposition, logic, algorithms, and data representation ➢ analyse problems in computational terms through practical experience of solving such problems, including designing, writing and debugging programs. ➢ understand the components that make up digital systems and how they	• Foundational Concepts Topic 6: Learning to program is a core component of a computer science course. Students should be competent at designing, reading, writing and debugging programs. They must be able to apply their skills to solve real problems and produce readable, robust programs. Topic 4: Most computer applications in use today would not be possible without networks. Students should understand the key principles behind the organisation of computer networks. • Key vocabulary decomposition, subprogram interface, turtle graphics, Cartesian coordinates, iteration, selection, repetition, turtle pen, pen colour, pen size, begin_fill, end_fill, LAN, WAN, network, protocol, IP address, bandwidth, latency, bits per second (bps), wired connectivity, wireless connectivity, fibre-optic, copper cable, Wi-Fi, Bluetooth, RFiD, Zigbee, NFC, network topology, bus, star, mesh, POP, NAP, backbone, router • Commentary ✓ Decompose a larger problem and design modular turtle programs ✓ Use turtle graphics with coordinates, loops, and conditional statements ✓ Change turtle pen colours and sizes, and apply fill functions for closed shapes ✓ Use subprograms to structure code effectively in a final turtle challenge ✓ Explain LANs, WANs, and the role of protocols and IP addresses ✓ Understand how bandwidth and latency affect network performance ✓ Compare wired and wireless technologies, including copper vs fibre-optic ✓ Describe wireless communication methods including Wi-Fi, Bluetooth and NFC
9-Jun	B	34		
16-Jun	A	35		
23-Jun	B	36		
30-Jun	A	37*		
7-Jul	B	38*		

14-Jul	A	39*	communicate with one another and with other systems	✓ Identify and compare network topologies and internet infrastructure components • **Assessment –** Informal assessment Sample assessment materials will be available to help prepare learners for the formal assessment
(Total: 190 Days)				

*Weeks 37-39 are **likely** to be impacted by college visits, year rewards trip, sports day and work experience week.