Compute-IT Scheme of Work

This Scheme of Work document provides a brief overview of the Challenges (large problem-solving activities) and content that each Unit of the Compute-IT course is designed to deliver, along with notes and teaching time required, and how the Units cross reference to the broad strands of Computing and IT outlined in a Progression Pathway chart that has been written by authors, reviewed and approved by Computing At Schools and published by Hodder Education.

(The Progression Pathway chart can be freely downloaded as two versions at www.hoddereducation.co.uk/compute-it. Direct download links are:

https://www.hoddereducation.co.uk/media/Documents/ICT/Progression_Pathways_Assessment_Framework_V1-2.pdf - for a version that is arranged by content area and https://www.hoddereducation.co.uk/media/Documents/ICT/CPP-V2.pdf - for a version that is arranged by National Curriculum strand)

The comprehensive teacher notes that are provided for every lesson of every unit of the Compute-IT course provide far greater detail for teachers in terms of the more granular learning objectives that contribute to teacher assessment of student attainment and guidance on how to assess the work students produce in order to come to judgements about their level of achievement and to measure their progression.

Compute-IT 3 (Year 9)

Unit Name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 1: Cracking the code: binary characters, cyphers and encryption	This unit aims to provide students with an understanding of cyphers, encryption and decryption, and the different methods used to encrypt throughout history.	The challenge is to invent a cypher or secret code that only one other person understands – the student is a secret agent who needs to send an encrypted classified message, via email, to a fellow-spy in another country.	This unit covers the NC bullets: 'Understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits.' 'Understand a range of ways to use technology safely, respectfully, responsibly and securely, including protecting their online identity and privacy; recognise inappropriate content, contact and conduct and know how to report concerns.' 'Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems.' 'Understand and use binary digits, such as to be able to convert between binary and decimal and perform simple binary addition.' 'Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures such as lists, tables or arrays; design and develop modular programs that use procedures or functions.' 'Understand simple Boolean logic (such as AND, OR and NOT) and some of its uses in circuits and programming.' The first Lesson revisits binary coding and shows its importance in the digital world, considering ASCII and other coding systems. By the end of the lesson **all** students must understand that there are different systems for representing text in binary form; and be able to use a spreadsheet that converts decimals into binary and ASCII. **Most** students should be able to understand the different binary systems that are used, and create a spreadsheet that converts decimals into binary and ASCII. **Some** students will be able to explain the differences between and the uses for the different character coding systems, and develop a spreadsheet to carry out binary conversions into various formats and vice versa. Lesson 2 covers encryption and cyphers through time, and the relationship with online privacy and security.	Algorithms Information Technology Programming and development Data and data representation	6 weeks @ 45 mins to 1 hour per week.

Unit Name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 1: Cracking the code: binary characters, cyphers and encryption – *continued*			By the end of this lesson **all** students should understand that encryption is a way of keeping data secure from unauthorised users, and that data can be encrypted and decrypted using cyphers. **Most** will be able to identify common features about different cyphers used throughout history, and know that without the key to decrypt cypher text we cannot convert it back into plaintext. **Some** will be able to explain the mathematical principles behind some of the major cyphers in history. Lesson 3 is about understanding and developing students' own cyphers. By the end of it, all students will be able to use a simple cypher to convert plain text to cypher text and back again. Most students will develop a more complex cypher using a substitution or transposition method. Some will develop a range of cyphers using a variety of methods. Lesson 4 leads on from the previous lesson, as students will be creating a spreadsheet to encrypt and decrypt data using one or more of the cyphers they designed during the last lesson. By the end, all students will be able to create a spreadsheet for a simple substitution cypher. Most will be able to create spreadsheets that will carry out substitution and transposition cyphers. Some will create spreadsheets for complex cyphers. Lessons 5 and 6 require students to write a program in a text-based programming language, using new techniques. By the end, all students will have created a simple cypher. Most will create a cypher using a text-based programming language. Some will create a ***complex*** cypher using a text-based programming language.		

Unit Name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 2: Representing sounds	This unit explores the way that computers store and execute binary information, including sound.	The student challenge is to advise a local record label which wants to open its own independent music store and allow users to stream music live to their mobile digital devices over 3G. They need a compression strategy that will allow them to keep the file size down without compromising on sound quality.	This unit covers the NC bullet: 'Understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits.' Lesson 1 introduces 'sampling' and streaming. By the end of the lesson, all students will understand about uncompressed file size and that a bigger file takes longer to download. Most will understand that a sound file can be displayed as a graph. Some will know different media formats and understand their typical file sizes. In Lesson 2 students will be finding out more about how sounds are captured and stored as binary data. By the end, all students must understand that sounds are stored as individual samples; and will know that having more samples means a better quality sound. Most will understand that an increased number of samples leads to more accurate sound reproduction; and will know that more samples lead to a bigger overall file size. Some will understand the connection between streaming, downloading and file quality. Lesson 3 covers lossy and lossless compression. By the end of it, all students should know that there are different formats for sound files; understand that reducing the file size will often affect sound quality; and understand the difference between lossy and lossless data compression. Some students will know that lossless compression techniques don't affect sound quality; and understand that lossless compression does not reduce the file size as much as lossy compression. Some will even be able to confidently evaluate the advantages of lossless and lossy compression.	Data and data representation Information technology	3 weeks @ 45 mins to 1 hour per week.

Unit Name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 3: Simple database tables	This unit is designed to provide students with an understanding of what is involved in building a database.	Nobody knows the subject of Variation XIII of Edward Elgar's ***Enigma Variations***, although there are clues. The student challenge is to discover who this piece of music is about by following the clues and constructing and interrogating a database.	This unit covers the NC bullets: 'Undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users.' 'Understand simple Boolean logic (such as AND, OR and NOT) and some of its uses in circuits and programming.' 'Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures such as lists, tables or arrays; design and develop modular programs that use procedures or functions.' Lesson 1 re-introduces the structure and purpose of databases. By the end of the lesson, all students should be able to explain the terms 'table', 'record' and 'field'; select relevant data for storing in a database; and create a table of data. Most will be able to explain the term 'key field'; extract relevant data from passages of text; and select the correct data type for a field. Some may be able to import data from a spreadsheet into a database. Lesson 2 explores Boolean operators. By the end of the lesson, all students should be able to use the Boolean operators AND, OR, NOT and wildcard operators; and be able to construct simple database queries using the database query wizard. Most students will be able to explain how the Boolean AND, OR, NOT and wildcard operators work; construct database queries using a combination of Boolean operators, using the database query wizard; and search a database efficiently. Some will be able to construct complex database queries using SQL. Lesson 3 deals with entering and checking data, then querying it to find the results you want. At the end of this lesson, all students will be able to construct simple database queries to find answers to specific questions using the database query wizard; they will also understand the concept of data redundancy and how to reduce it. Most students will be able to construct a series of database queries to find answers to specific questions using Boolean operators and the database query wizard. Some may be able to construct complex database queries to find answers to specific questions using SQL.	Programming and development Data and data representation Information Technology	3 weeks @ 45 mins to 1 hour per week.

Unit Name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 4: Searching	This unit covers searching and the different algorithms that can be used.	The student challenge is to create a spell checker.	This unit covers the NC bullets: 'Understand several key algorithms that reflect computational thinking (for example, ones for sorting and searching); use logical reasoning to compare the utility of alternative algorithms for the same problem.' 'Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures (for example, lists, tables or arrays); design and develop modular programs that use procedures or functions.' Lesson 1 opens the topic of searching, with linear searches. By the end, all students should be able to describe what is meant by a linear search. Most students will be able to describe the linear search algorithm. Some will be able to describe the limitations of the linear search algorithm. Lesson 2 develops the topic of linear searches and looks at the code behind them. By the end, all students should have followed the pseudocode for a program that performs a linear search. Most will have written a linear search program in a graphical programming language. Some will write a linear search program in a graphical programming language that can deal with items that are not present in the array or list. Lesson 3 is about arrays and lists, and binary searches. By the end of the lesson, all students should be able to describe what is meant by a binary search. Most will be able to code a binary search with suitable support. Some will independently code a binary search. Lesson 4 compares linear and binary searches. By the end of it, all students should be able to empirically compare linear and binary search. Most will be able to compare linear and binary search, describing which is best in a given situation. Some will sketch graphs representing the mathematically best, worst and average number of 'checks' needed for linear and binary search.	Algorithms Programming and development	6 weeks @ 45 mins to 1 hour per week.

Unit Name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 4: Searching – *continued*			Lesson 5 begins to tackle the Challenge, with the intention for it to be completed using a graphical programming language, such as Scratch, or a text-based programming language such as Python. By the end, all students should be able to apply the binary search algorithm to locating a name in a list. Most students will have created a program to locate a single name in a list using the binary search algorithm. Some will create a program that can locate or identify that a name is ***not*** present in a list using the binary search algorithm. Lesson 6 completes the spell checker challenge. By the end of this final lesson, all students should be able to describe how searching algorithms can be applied to the problem of creating a spell checker. Most students will have coded a spell checker to check a single word in Scratch. Some will code a simple spell checker program that checks prose in a text-based programming language.		

Unit Name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 5: Getting down and dirty withnetworks	This unit covers what networks are, then moves on to how to design and build a simple network. It explains protocol communication rules, SMTP, IMAP, POP, network topologies, IP addresses etc.	Students are challenged to build and test a network.	This unit covers the NC bullets: 'Understand computer networks including the internet; how they can provide multiple services, such as the world-wide web; and the opportunities they offer for communication and collaboration.' 'Understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems.' 'Understand and use binary digits, such as to be able to convert between binary and decimal and perform simple binary addition.' Lesson 1 picks up and continues the work on Networks in Compute-IT 2. By the end of the lesson, all students should appreciate that computers use a number of different protocols to achieve different tasks on a network. Most will be able to trace the origin and path of an email, and know the application layer presents the abstraction of direct application-to-application communication. Some will be able to make an informed decision on which protocol, POP3 or IMAP, is better for a given situation, and know how communication is passed through the layers of abstraction, across a physical network and back up again. Lessons 2 and 3, between them, take students into designing and building a simple network. By the end of the third lesson, all students should be able to access the MAC address of their digital device; convert a hexadecimal number into decimal; and describe three network topologies, the bus, the star and the ring. Most students will be able to look up the manufacturer part of a MAC address to find out who made the NIC; convert between decimal, hexadecimal and binary; know the advantages and disadvantages of three network topologies, the bus, the star and the ring; and design a basic network. Some may design and build a network.	Data and data representation Communication and networks	3 weeks @ 45 mins to 1 hour per week

Unit Name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 6: Client-side and server-side scripting	This Unit is all about webforms, HTML and Javascript, passwords, data protection and validation.	The student challenge for this unit is to build a web page that allows users to fill in a form to search for data within a database and to build a server-side script page that uses SQL to look for the data.	This unit covers the NC bullets: 'Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures such as lists, tables or arrays; design and develop modular programs that use procedures or functions.' 'Understand a range of ways to use technology safely, respectfully, responsibly and securely, including protecting their online identity and privacy; recognise inappropriate content, contact and conduct and know how to report concerns.' Lesson 1 starts by building a webform using HTML and Javascript. By the end, all students will be able to build a basic HTML form; use events to call a Javascript function; and write a basic Javascript validation function. Most students will be able to add some formatting to an HTML form to lay out the elements, and write more complex Javascript functions. Some may be able to write Javascript functions to validate complex data items, and use HTML to build a professional-looking web form. Lesson 2 deals with basic server-side scripting on a local web server. By the end, all students should be able to write basic server-side scripting, including outputting information to the screen; and understand where server-side and client-side scripting occur. Most students should be able to output submitted data to the web server using server-side scripting; and understand how data moves between a client server and a web server. Finally, some students may be able to use more advanced server-side scripting using loops or functions, and write custom HTML code using server-side scripting. Lesson 3 completes the challenge using a text editor, a web browser and a local web server. By the end of it, all students should be able to plan a basic database table structure, and create a database table. Most students should be able to plan a database table structure considering data types and data sizes, and know how to insert data into a database table using server-side scripting. Some students may be able to retrieve data from a database table using server-side scripting.	Communication and networks	3 weeks @ 45 mins to 1 hour per week

Unit Name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 7: Digital circuits	This unit introduces students to circuit building and logic gates.	Students are challenged to build their own digital security alarm.	This unit covers the NC bullets: 'Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems.' 'Understand simple Boolean logic (such as AND, OR and NOT) and its use in determining which parts of a program are executed.' 'Understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems.' 'Understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits.' Lesson 1 employs some simple and inexpensive components in a practical, enjoyable activity including switches and logic gates. By the end, all students should understand the role of a switch as a human interface with an electrical circuit; be able to complete a truth table for AND and OR gates; and understand that the behaviour of electrical circuits is reliable, predictable and logical. Most students will understand the function of an SPDT switch in a stairway circuit; be able to explain the role of the NOT gate as an inverter and complete the truth tables for the NAND and NOR gates; and understand the term Boolean logic. Some students may correctly wire a DPDT switch and complete a truth table for a string of linked gates. Lesson 2 continues the practical activities, building a breadboard circuit. By the end of the lesson, all students will be able to build a NOT gate circuit; understand the function of a NOT gate as an inverter; and know that computer circuits are built from transistors. Most students will be able to successfully build a NOT gate circuit without support; successfully build an AND gate or an OR gate circuit; understand that the output from a gate depends on the pattern of transistors in its circuitry; and understand the role of the transistor as a current-operated switch. Some will be able to successfully build an AND gate and an OR gate circuit without support.	Data and data representation Hardware and processing	4 weeks @ 45 mins to 1 hour per week

Unit Name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 7: Digital circuits – *continued*			Lesson 3 includes activities with integrated circuits. By the end of this lesson, all students should be able to identify the main parts of an integrated circuit, and know that integrated circuits can perform a range of different functions including keeping time and adding up. Most students will be able to complete the truth table for a half adder; explain how the half adder works; and apply Boolean logic to predict the output from a string of gates. Some will be able to complete the full adder truth table. In Lesson 4, the Challenge is undertaken, with the use of a latch circuit and a demonstration of how computers can store data. By the end of this lesson, all students should know that computers can store data in memory; be able to build a security alarm circuit; and understand the role of the pressure pad as an input device. Most students will understand that a latch is a circuit with two possible stable states; and will be able to successfully build and operate a security alarm circuit without support. Some students will understand and explain the limitations of Moore's law in the future, and be able to explain how a latch circuit works.		

Unit Name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 8: Computer architecture	This unit covers compilers, interpreters and translators, the fetch-decode-execute cycle, and the Von Neumann and Harvard architectures.	The student challenge is to use knowledge and research skills to compare different computer architectures.	This unit covers the NC bullets: 'Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems.' 'Understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits.' 'Understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems.' Lesson 1 is about spoken languages and programming languages, and shows the role of interpreters, compilers and translators in both. By the end of the lesson, all students should understand that translators convert high level programming languages to machine code; know how compilers translate high level programming languages into machine code; and know how interpreters translate high level programming languages into machine code. Most students will understand how interpreters and compilers can be used together by programmers; be able to simulate the actions of either an interpreter or a compiler; be aware of the use of library routines in programming; and be able to weigh up the advantages and disadvantages of compilers and interpreters. Some students will understand the benefits of using library routines when creating a program. In Lesson 2 the fetch–decode–execute cycle is re-introduced using Little Man Computer, and the Von Neumann and Harvard architectures are examined. By the end of the lesson, all students will be able to state and describe the three phases of the fetch–decode–execute cycle, and know the difference between the Von Neumann and the Harvard architectures. Most students will be able to design and simulate basic arithmetic in assembly language, and know the key parts of the CPU architecture such as the program counter, accumulator etc. Some students will be able to explain the advantages and disadvantages of both types of architecture.	Programming and development Hardware and processing	2 weeks @ 45 mins to 1 hour per week

Unit Name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 9: Creating an app to solve a problem	This Unit tackles apps and app development – what are the different types and how are they built?	Students use all the knowledge, skills and experience they have developed over the entire KS3 course to create an app to solve a problem that is on their doorstep, at school or at home. Could an app help solve that problem?	This unit covers the NC bullets: 'Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems.' 'Develop and apply their analytic, problem-solving, design, and computational thinking skills.' 'Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures such as lists, tables or arrays; design and develop modular programs that use procedures or functions.' 'Undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users.' Lesson 1 introduces apps - inspired by a program developed and run by Apps for Good in 2013-2014. By the end of it, all students should be able to identify if an app is a web app or a native app. Most students should be able to identify some roles in a team project, and identify if an app is a hybrid app. Some students will be able to identify roles in a team project and the consequences of changes. Lesson 2 pushes brainstorming, and the work of designing and developing an app, further forward. By the end, all students should be able to identify some of the key elements of a problem. Most students should be able to evaluate ideas using a range of techniques. Some students will even devise an elevator pitch to explain an idea. In Lessons 3 and 4, students are encouraged to scope out their ideas further. By the end, all students should be able to identify some factors that affect the viability of a new app, and design a method to collect data from potential users of a new app. Most students will be able to justify the viability of a new app; analyse data collected from potential users of a new app; and create an algorithm to describe a new app.	Algorithms Information technology Programming and development	6 weeks @ 45 mins to 1 hour per week

Unit Name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 9: Creating an app to solve a problem – *continued*			Some students will create a revised algorithm based on feedback from research into a new app; and be able to justify the solution to a problem in terms of feedback from research. Lessons 5 and 6 complete the challenge of building an app. By the end of Lesson 6, all students must be able to describe the objectives of the app; create a screen layout and be able to code some part of the app; and be able to test the app on an appropriate end user. Most students will be able to accurately describe and prioritise objectives for the app; create a suitable screen map and some wireframe diagrams; code a working app; and test the app on an end user and respond to the feedback. Some students will be able to create a screen map and wireframe that fully describes the app; code a fully working app; and modify the app in light of feedback.		