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 2 (Year 8)

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 1: Operating Systems	This unit aims to provide students with an understanding of the different types of Operating Systems that exist and how they are designed to work in different contexts	Create a poster to explain the similarities and differences between common operating systems to help someone decide which one is the best for them.	This unit covers the NC bullets: 'Understand the hardware and software components that make up computer systems and how they communicate with one another and with other systems' '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.' The first Lesson considers the following topics: What is an operating system? (Looking at layers of abstraction); Different types of operating system; and Grouping operating systems. By the end of the lesson **all** students must be able to describe what is meant by an operating system and should know the names of a range of operating systems. **Most** students should be able to explain the main features of an operating system and identify some common similarities and differences between operating systems. **Some** students will understand the difference between proprietary and open source operating systems. Lesson 2 asks students to rank statements about operating systems and reintroduces the challenge to produce a poster. By the end of the second lesson **all** students must be able to describe examples of different operating systems. **Most** should be able to identify positive and negative features of two or more operating systems, and **some** will be able to compare the strengths and weaknesses of operating systems, choosing which is most suitable for a given scenario. They will produce a poster that can be self- or peer-reviewed and / or assessed by the teacher.	Hardware and Processing Information Technology	2 weeks @ 45 mins to 1 hour per week.

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 2: CMD, the command line	The aim of this unit is to provide students with an understanding of the use of basic command prompts to manipulate files and folders, and write a batch or shell script.	The student challenge is to play the role of someone who has started a new job with GCHQ (Government Communication Headquarters) as a forensic analyst. Their first task will involve showing others how to use computers to manage files and folders, before they move on to more advanced file handling techniques that can be used in digital investigations.	This unit covers the NC bullets: 'Use technology purposefully to create, organise, store, manipulate and retrieve digital content.' 'Understand the hardware and software components that make up networked computer systems.' For this Unit you will need to be familiar with the CMD line as well as file management commands in the CMD line. It is worth noting that Mac and Linux use the same commands because they are both flavours of Unix and – mostly – POSIX standards compliant. Lesson 1 considers older people taking their first steps on the internet, and good methods of file management. By the end of Lesson 1 **all** students must be able to create, move and rename files and folders; be aware of at least one method of copying and pasting files; and understand that sensible file and folder names make it easier to find work later. **Most** students should be able to explain how to manipulate files and folders in clear, simple terms and demonstrate understanding of different methods for cutting, copying and pasting. **Some** students will be able to identify strategies for naming folders and nesting folders (folders within folders); appreciate the advantages and disadvantages for different methods of cutting, copying and pasting; and be aware of file extensions and their purpose. Lesson 2 introduces a reverse crossword and asks students to familiarise themselves with the command line prompt. By the end of this lesson **all** students must able to type at least one instruction at a command prompt, and understand that there are two ways to manage files and folders. **Most** students should be able to move, copy, rename and delete files and folders at a command prompt; and also be able to identify advantages and disadvantages of both graphical and command line methods for managing files and folders.	Data and data representation Hardware and processing	3 weeks @ 45 mins to 1 hour per week.

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 2: CMD, the command line – *continued*			**Some** will understand the link between the GUI and text-based commands that execute within an operating system, and suggest situations where the command line would be preferable to a GUI for file manipulation. By the end of Lesson 3 all students must be able to create a batch file or shell script that will run at least one command, and understand that it might be quicker to run a script than to run commands individually every time. **Most** students should be able to move, copy, rename and delete files and folders in a script, and identify advantages and disadvantages of writing scripts. **Some** will understand how to select files according to criteria, and suggest situations where the command line would be preferable to a GUI for manipulation.		

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 3: Binary	This unit is designed to provide students with an understanding of how binary and binary addition are designed to work.	Students are challenged to create a video tutorial to help students of a similar age learn binary and binary arithmetic.	This unit covers the NC bullets: 'Understand and use binary digits, such as to be able to convert between decimal and perform simple binary addition.' 'Create, reuse, revise and repurpose digital artefacts for a given audience, with attention to trustworthiness, design and usability.' It is important that you understand how the binary number system and the decimal number system relate to each other. By the end of Lesson 1 **all** students will know the value of each bit of an 8-bit binary string; and be able to convert a decimal number to binary. **Most** students will be able to convert a decimal number to binary (1-255); be able to convert a binary number to decimal (1-255); and be able to count in binary to 8 bits. Some students will show an understanding of multiple methods of converting decimal to binary. By the end of Lesson 2 **all** students will be able to add two 4-bit binary numbers together; understand that binary numbers are held in 8-bit fixed bit strings on computers; and be able to explain how to convert binary to decimal and vice versa. **Most** students should be able to add two 8-bit binary numbers together; to add and convert binary numbers into 8-bit fixed bit strings; and to explain how to add two binary numbers together. **Some** students will be able to recognise and explain binary overflow when adding two 8-bit binary strings. By the end of Lesson 3 **all** students will produce a tutorial on an aspect of binary arithmetic; and evaluate and give feedback on another person's work using a guide. **Most** students will be able to produce a video tutorial on an aspect of binary arithmetic. **Some** students will in addition be able to produce a tutorial on what binary overflow is and when it occurs, and make adjustments to the work based on the feedback received.	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: Instruction Set Design	Using a chosen programming language, this unit challenges students to think about different aspects of programming to solve a 'real-world' problem or challenge.	The student challenge is to program a 3-bit rover robot to explore efficiently a planet far from Earth, avoiding obstacles it meets along its way.	This unit covers the NC bullets: 'Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into small parts.' '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.' '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 is about programming a robot with the most efficient instructions. By the end of Lesson 1 **all** students will be able to record a sequence of instructions of a given instruction list; and identify different routes to a destination and find a route with the least number of instructions. **Most** students should be able to discuss the trade-off between a route with a limited number of instructions and a route with a less limited range of instructions; understand that sensors are used to collect inputs in order to make decisions; and know that the more instructions that are used, the more memory is needed. **Some** students will be able to identify the best route from a range of algorithms. Lesson 2 uses Scratch to simulate a robot's instruction set. By the end of this lesson **all** students will be able produce a set of algorithms for basic sprite movement; identify instructions for a 2-bit instruction set; create a set of instructions to move an object in three directions; and identify instructions to solve a problem. **Most** students should be able to create procedures for a 2-bit instruction set; identify methods to navigate around given problems with constraints; and select 2-bit instructions and use them in a suitable sequence. **Some** students could develop algorithms that use binary representations of instructions as input.	Algorithms Programming and development	3 weeks @ 45 mins to 1 hour per week.

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 4: Instruction Set Design – *continued*			Lesson 3 is where students will complete the challenge for this Unit – programming a 3-bit rover robot to explore efficiently a planet far from Earth, avoiding obstacles it meets along its way. By the end of this lesson, **all** students will know that instructions in a set can be given different bit patterns; and adjust the movements that a robot can make so that it can respond to obstacles. **Most** students should understand that when an instruction set is enlarged, every instruction requires more bits; be able to produce algorithms and procedures to enable a robot to navigate obstacles in its path; and understand the scale of instruction sets in the context of modern-day storage devices and memory. **Some** students will be able to develop algorithms and procedures that enable a robot to deal with obstacles effectively; and know how decreasing the number of instructions used to solve a problem can have a significant impact when scaled to a large environment.		

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 5: Programming using selection statements and Boolean expressions	This unit builds on previous work and provides students with the basis for using Boolean statements and expressions	Students are challenged to program a robot that appears to be intelligent – a robot that can successfully navigate around a maze all by itself.	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 and some of its uses in circuits and programming; understand how numbers can be represented in binary and be able to carry out simple operations on binary numbers.' '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.' In Lesson 1 students will be using a graphical programming language to control a robot around a maze and a circuit. They will be using selection and decomposition. At the end of this lesson, all students will understand that problems need to be broken down into more manageable parts; and how selection statements can be used to provide alternate paths in a solution. They will also be able to sequence a set of selection statements. Most students will be able to combine selection and the Boolean operators AND and OR; and be able to use variables as a method of recording incremental changes. Some students will be able to write algorithms that use selection statements with Boolean operators. Lesson 2 includes using nested selection and Venn diagrams, and introduces students to the NOT operator. At the end of this lesson, all students should be able to nest 'if' statements to enable two inputs to be checked; understand that Venn diagrams can be used to demonstrate graphically nested selection sequences; and be able to use the NOT operator. Most students will be able to represent nested 'if' statements using suitable diagrams; and be able to use AND, OR and NOT operators. Some students will be able to combine selection statements and Boolean operators including NOT. Some will be able to review the solution of their algorithm and make adjustments if necessary.	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 5: Programming using selection statements and Boolean expressions – *continued*			Lesson 3 builds on and extends what has been learned in Lesson 2. By the end of Lesson 3, all students will be able to use 'if' statements with AND; understand that Boolean logic combines Boolean operators; and be able to identify the outputs of a truth table for a group of connected Boolean operators. Most students will be able to combine 'if' selection and Boolean operators; be able to use 'if else'; and will understand how different forms of 'if' statement can affect the organisation of programs. Some students may be able to combine a range of selection statements and Boolean expressions in a program; and understand the best type of 'if' selection statement to use. Lesson 4 explores loops and patterns. By the end of it, all students will be able to identify where loops can be used to repeat commands; and understand that there are a variety of different loops that are used in different situations. Most students will be able to program using three different types of loop; analyse situations to identify repeated sections that can be reused; and understand how flowcharts can be used to model repetition. Some may be able to carry out detailed analysis of where loops can be used within a program. Lesson 5 concentrates on algorithms and mazes. By the end of it, all students will understand that there are a range of standard algorithms that have been designed for mass use, and they will know that two algorithms that solve the same problem may not be the same. Most students will know at least two algorithms for navigating a maze and how they differ from one another; create a Random Mouse algorithm; and create a program that uses the Wall Follower algorithm to navigate a robot through a maze. Some students will be able to implement at least two different algorithms for navigating through a maze. In Lesson 6, students will complete the challenge for the Unit – programming a robot to navigate a maze. By the end of this lesson, all students will be able to adjust algorithms to deal with exceptions. Most will be able to create a procedure to deal with an exception, and understand how a procedure can be represented in a flowchart.		

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 6: Connecting to the internet	This Unit is all about connecting to the Internet.	The student challenge is to split a message into data packets similar to the TCP/IP protocol.	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.' Lesson 1 deals with how people connect to the Internet, past, present and future. By the end of it, all students will know and understand the common ways to connect to the internet, and understand the role of the ISP. Most students will understand the relationship between LANs and WANs, and be able to research internet usage and the popularity of different ways to connect to the internet. Some students will understand the way in which telephone lines are used to transmit digital data. Lesson 2 concentrates on packets of data, and IP addresses. By the end of this lesson, all students will understand how data streams generated by computer applications are split into packets; will know and understand the role of routers; and will be able to distinguish between circuit switching and packet switching. Most students will be able to appreciate the advantages of packet switching for most connections, and be able to remember the format of IP addresses. Some will be able to understand the connection between IP addresses and domain names. Lesson 3 centres around the layers of data that the internet is made from. By the end of it, all students must know the four TCP/IP layers; understand that the four TCP/IP layers are independent of each other; and understand that data can become corrupted. Most will be able to identify key features of the four TCP/IP layers, and calculate the parity bit. Some will be able to describe metadata layering for a packet of data.	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: Sorted!	This unit introduces students to the idea of sorting. They will examine how computers and humans cope with sorting, and experiment with running sorting algorithms on different sets of data.	Students are challenged to write a program for a high-score table for a computer game.	This unit covers the NC bullets: 'Understand several key algorithms that reflect computational thinking such as 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 such as lists, tables or arrays; design and develop modular programs that use procedures or functions.' Lesson 1 considers sorting in ascending and descending order, and introduces BogoSort and Swap Sort. By the end of it, all students must be able to explain what is meant by a sorting algorithm. Most students will be able to explain how the size of a data set affects the time it takes for an algorithm to sort data. Some students will make amendments to a given sorting algorithm, for example switching it to sort from ascending to descending order. Lesson 2 moves on to Bubble Sort. By the end of it, all students must be able to describe how the Bubble Sort algorithm works. Most will be able to apply the Bubble Sort algorithm to a set of data. Some will be able to explain what limitations Bubble Sort has. Lesson 3 extends lesson 2 with more on Bubble Sort. All students, by the end of it, will be able to code the Bubble Sort algorithm with structured guidance; will use iteration to move through a list one item at a time; and swap the contents of two variables via an intermediary variable. Most will be able to code the Bubble Sort algorithm with limited additional guidance; they will also know that pseudocode is used in program design and is a system of writing that resembles a simplified programming language. Some students will be able to code the Bubble Sort algorithm without the need for any additional guidance, and be able to use pseudocode in program design.	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 7: Sorted! – *continued*			Lesson 4 moves on to Selection Sort. By the end of it, all students will be able to describe how the Selection Sort algorithm works. Most will be able to apply the Selection Sort algorithm to a set of data. Some will be able to explain whether the order data starts in affects the speed of Selection Sort. Lesson 5 continues with Selection Sort. At the end of it, all students will be able to code the Selection Sort algorithm with structured guidance; write code to find the smallest item in a list; and understand that 'nesting' is the process of placing a section of code within another section of code. Most students will be able to code the Selection Sort algorithm with the help of a pseudocode example, and be able to use 'nesting' in Selection Sort. Some will be able to code the Selection Sort algorithm independently. Lesson 6 compares several sorting algorithms to see which is best for which purpose. By the end of this lesson, all students will be able to experiment with sorting algorithms to compare them. Most students will be able to use the results of their experiments to comment on the performance of sorting algorithms. Some will be able to select the best sorting algorithm for a given set of data.		

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 8: How to make a computer appear smart	The theme of this unit is an understanding of what intelligence is, involving the Turing Test – the test of a machine's ability to exhibit intelligent behaviour equivalent to that of a human.	The student challenge is to make a computer appear intelligent by holding a conversation with a human.	This unit covers the NC bullets: 'Use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs.' 'Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems.' 'Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts.' '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 considers intelligence and discusses the concept of thinking. By the end of it, all students will be able to describe some aspects of intelligence; understand that there are debates about whether or not a machine can display intelligence; and describe some examples of machines demonstrating behaviours that could be considered intelligent. Most students will be able to suggest ways in which humans can make use of machines that demonstrate seemingly intelligent behaviour. Some will be able to describe how existing artefacts could be improved to make them appear more useful and therefore more intelligent. Lesson 2 gets students started with Python programming and discusses the role of input and output. By the end of it, all students will be able to make use of basic input and output statements in the chosen text-based programming language; create and run a simple program in a text-based programming language; understand what is meant by a bug; and be able to debug their program. Most students will be able to use conditions such as 'if else'; plan and carry out a test plan, and use variables to store inputs. Some students may also show awareness of the need to test code for various exceptions, know the different types of test data, and use them in the test plan.	Programming and development Hardware and processing Information technology	6 weeks @ 45 mins to 1 hour per week

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 8: How to make a computer appear smart – *continued*			Lesson 3 extends Lesson 2 by getting students writing and editing in a text-based programming language. At the end of it, all students will be able to start, write, edit, save and run a simple program; write a loop in a text-based programming language; write and call sub-programs in a text-based programming language; and understand what iteration is. Most students will understand the difference between, and uses of, both top-tested loops such as 'while end while' loops and post-tested loops such as 'repeat until' loops; they will also know that a procedure can be used to hide detail within a sub-program and that this is called procedural abstraction. Some students will understand the difference between a 'while end while' loop and a 'for' loop which uses a counter; and design and write nested modular programs. Lesson 4 is all about data types. By the end of this lesson, all students will understand basic data types; understand what variables are and their limitations; and know how to initialise variables and arrays. Most students will be able to access data by index from an array or list, and understand how and why values are data typed in programs. Some students will know the difference between static and dynamic data typing, and the difference between different data structures. Lesson 5 introduces the concept of random numbers and randomness generally. By the end of the lesson, all students will understand how to generate random numbers in a text-based programming language; be able to extract data randomly from a data structure such as an array or list; and be able to plan and carry out a test plan. Most students will understand a means of pairing data from two separate arrays/lists; identify suitable test data and use it as part of a test plan; show awareness of the need to test code for various exceptions; and suggest ways of trapping some errors. Some students will determine ways of recording which items have been extracted from a data structure; determine that the use of data structures such as dictionaries and tuples can be more appropriate than lists/arrays to solve some problems; and understand that data structures can be 'nested' in lists of lists or multi-dimensional arrays.		

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 8: How to make a computer appear smart – *continued*			Lesson 6 considers whether computers can take the place of humans, and introduces 'Eliza' and the Turing Test. By the end of it, all students will be able to create a program that responds variably to inputs in a related but random way, and understand what is meant by the term 'natural language'. Most students will be able to write a program that produces reasonably believable responses to human inputs, and create a program that asks follow-up questions. Some students will be able to suggest ways of improving a simple conversational program; understand the nature of the Turing Test for intelligence; and understand the purpose of different data structures.		

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 9: Recursive Patterns	This Unit tackles the difficult topic of recursion, using songs and stories.	Students become pattern detectives and investigate some of the ways in which patterns can be created in mathematics, art and music using computers. They will use their new-found knowledge to write a program that enables a computer to output a recursive pattern.	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.' '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 considers recursion in 'One Man went to Mow' and other contexts. At the end of the lesson, all students will be able to identify patterns within songs (what changes and what stays the same; understand and be able to explain the basic concepts of recursion; and understand and be able to explain the key concepts of decomposition and iteration. Most students will be able to create an algorithm based on a decomposed problem, and identify how a pattern is recursive. Some students will be able to write algorithms for a recursive or iterative pattern. Lesson 2 uses a graphical programming language to create a program to 'sing' a recursive song. By the end of the lesson, all students will be able to pick out the parts of a pattern that are changing and the parts of a pattern that are staying the same; create image/sound files required by a program that will 'sing' a song; and interpret an algorithm. Most students will be able to convert an algorithm into a program, and create an iterative program. Some will be able to create a recursive program. Lesson 3 considers the Golden Ratio, the Fibonacci series and the Lucas series. By the end of the lesson, all students will be able to follow simple mathematical procedures to generate a number sequence, and create a program to output the Fibonacci Series. Most students will be able to define custom functions in a text-based programming language. Some students will create a recursive program in a text-based programming language. Lesson 4 encourages students to complete the challenge – to create and document a recursive program. By the end of it, all students will be able to design an algorithm to describe a program to output a recursive sequence; and program in a text-based programming language. Most students will be able to comment on a program, and test and debug a program. Some may create a report explaining the design, implementation and functionality of a program, and write a series of recursive programs to output various sequences.	Algorithms Information technology Programming and development	4 weeks @ 45 mins to 1 hour per week