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_VI-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 1 (Year 7)

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 1: Under the hood of a computer	This unit provides a brief outline of the history of computing; practical study of components that make up a computer; inputs, processing and outputs; data and binary; bits, bytes and megabytes.	Pupils learn to 'think' like a computer, and understand how computers process data.	You will be providing a very brief outline of the history of computing and the electronic computer. There is a useful external resource for this at www.docstoc.com/docs/7805995/History-of-Computers---PowerPoint-4. This is followed by a practical study of computer hardware to introduce students to the components that make up a computer. If old hardware is not available, the same activity can be carried out with good quality images of the components. You will be asking the students to research the functions of the component parts, so it is important that you are able to recognise each component and give an overview of its function. This link will take you to a useful source of background information: www.tutorialspoint.com/computer_fundamentals/index.htm At their most basic level, function machines take a number (such as 4) as INPUT, apply a process (such as 'multiply by 3') and give another number as OUTPUT (in this case, 12). They can apply more than one process (such as 'add 7 and then multiply by 3', producing an output of 33). They can also be run in reverse (for example, you could input 33, divide by 3 and subtract 7, producing an output of 4. There is no stored 'program' in a function machine; there must be manual intervention at the beginning to repeat the calculation with different input data. The role of Bletchley Park in shortening the Second World War and the part played by Colossus in breaking codes is a story that every student learning about computing should be aware of. BBC films on the subject can be found at: www.bbc.co.uk/history/places/bletchley_park Students will be taught about electrical circuits. Morse code is mentioned as a method of using electrical signals to send messages and using codes to represent characters. Morse code uses long and short signals and spaces to represent characters, whilst digital computing uses simple, yet elegant, on/off binary patterns. It is helpful to understand the similarities and differences between the two systems and why the more complex Morse code system is not a suitable code for processing data electronically (because, in addition to the two states of 'on' and 'off', Morse code contains a third factor that is required to distinguish between a dot and a dash – time).	Hardware and processing Data and data representation	2 weeks @ 45mins to 1 hour per week.

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 2: Think like a computer scientist	This unit introduces students to computational thinking.	Students are challenged to create an emergency evacuation plan for their school.	Students should understand that computational thinking is not 'thinking like a computer' but thinking about problems (and the world) in terms of the processes going on, the data available, and the steps that need to be followed in order to achieve a goal. They should understand that while human beings can use context and common sense to interpret instructions, can ask for clarification, and can act on their own initiative, computers cannot do any of these things. They should also understand that humans are best employed doing creative and innovative things, while computers are best used for repetitive tasks that require speed and precision. This is the first time students will meet the term in this course, so you will apply a simple version of computational thinking that can be broken down into a small number of discrete activities. The first lesson covers decomposition and algorithms. These terms refer to the process skills involved in solving problems by thinking logically. They will be revisited in a number of subsequent units, when the concepts touched on here will be explained in much greater depth. The emphasis at this stage is on having fun and the activities have been designed to make the subject as playful and engaging as possible.	Algorithms	3 weeks @ 45mins to 1 hour per week.

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 3: Drawing and manipulating Shapes	This unit is designed to provide students with an understanding of the relationship between computer science and shape/ patterns in order to be able to write algorithms in a range of computer programming languages to draw basic shapes and design artworks.	The students' challenge is to write a program that creates an artwork based on drawing and positioning shapes found in Celtic or Islamic art.	You will be getting students to appreciate the link between maths, art and computer science, particularly the creativity and imagination required to create works of art and computer programs based on both artistic and mathematical concepts. Students will understand simple algorithm design and the importance of being able to identify the important ideas (abstraction) and breaking down the problem into manageable units (decomposition). Students will also be introduced to repetition (iteration) as one of the key constructs in programming. You will help them discover how to design algorithms for some basic shapes. You will be getting students to write programs to draw shapes they have defined using algorithms. It is important not to confuse algorithms and code. An algorithm is a set of rules that defines a solution. It is possible to write an algorithm without using iteration and implement it in code. However, for the purposes of this unit, writing the algorithm using iteration should follow through into the code. This is followed up in Lesson 3 with a practical session that requires students to code their artworks.	Algorithms Programming and development	3 weeks @ 45mins to 1 hour per week.

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 4: Creating an animation	This unit requires the students to think about and create algorithms, so you will need to be comfortable with algorithms and the need for precision in framing instructions. Throughout we have supplied resources based on version 2.0 of Scratch, but you can use other graphical programming languages, such as BYOB, Alice or AppInventor.	Students are challenged to program an animation to entertain an audience by recreating a dance routine from a music video using programming techniques such as sequences, iteration, procedures, selection and variables.	Students will be using a graphical programming language to recreate a dance sequence. They are asked to take and use images of themselves in various positions to recreate these moves. You will need to show them how to upload these images for use in your chosen graphical programming language. You will also need to be able to demonstrate the basic constructs within the programming language, including wait commands and 'forever if' loops. Note that some students might not wish to use photographs of themselves, and suitable images could be provided for them to use. Students program their dance animation using more than one dance sequence, and need to become familiar with repeating, selecting, iterating and procedures in your chosen graphical programming language.	Algorithms Programming and development	3 weeks @ 45mins to 1 hour per week.

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 5: The foundations of computing	By understanding how computers have developed, students are encouraged to not only create programs to carry out arithmetic calculations, but to 'think' like a computer in order to so.	Computers simply follow the instructions given to them, so we need to convert the instructions given by humans into something the computer can understand. Students are challenged to write a program to carry out simple arithmetic calculations in a language the machine can understand and to think like the machine in order to do this.	This unit outlines how computers have developed from the basic calculators and machinery used to solve problems, to programmable, general-purpose computers, and some of the key people involved in that process. It goes on to cover machine languages and assembly languages. Students need to be fully aware that the CPU works with binary instructions and that the assembly language is little more than a short-cut to writing programs in binary. The assembly language used in this lesson is based on Little Man Computer. The unit concludes with coverage of how the CPU handles program instructions and data. Students are asked to consider more complex programs and to dry run these as a group.	Algorithms Programming and development Hardware and processing	3 weeks @ 45mins to 1 hour per week.

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 6: How the web works	This unit provides an opportunity to look at the way in which the web works technically, and cover the issues of reliability and e-safety.	Students are challenged to research efficiently and effectively three programming languages named after famous people, understanding how searches work and how to evaluate the reliability of results.	This unit starts with a lesson focused on getting students to appreciate the power and complexity of interlinked content on the web and to get a feel for the way navigation works. As long as you are a web user you shouldn't have too many problems, although understanding the folder–subfolder structure of websites and a basic understanding of HTML tags would be useful. W3 schools (www.w3schools.com/html) provide some useful tutorials. The following lesson is explorative and asks students to think critically about the content of websites and raises issues about e-safety. You should be aware of the issues raised and be prepared to discuss e-safety and the relative accuracy and reliability of websites. In the final lesson a basic understanding of how weblinks work is required. You also need to be familiar with Boolean operators (AND, OR and NOT), as well as the way Venn diagrams can be used to help students understand how they work.	Communication and networks	3 weeks @ 45mins to 1 hour per week.

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 7: Web page creation from the ground up	This unit provides students with the challenge of creating and uploading web pages.	The student challenge is to design and code a web page and upload it to a server.	This unit provides students with the challenge of getting to grips with HTML, considering things such as accessibility, style sheets, user interface design, embedding multimedia and uploading the web content to a server. We recommend that you have an HTML 5 compliant web browser installed on your system. If you are not sure about your browser, go to: http://HTML5test.com/. It is possible to do this entire unit on a tablet device, but there is some keyboard inputting required. You will need to know how to edit a web page. A quick way to do this is to use https://goggles.webmaker.org. Once you have installed X-Ray Goggles on your bookmarks bar, you can then click it to edit any page. Familiarise yourself with Mozilla Thimble (https://thimble.webmaker.org) or an alternative HTML coding package such as Blue Griffon (www.bluegriffon.org), which is open source, or even Microsoft Notepad. Please do not choose a package that does all the HTML coding in a WYSIWYG interface because you want students to learn the coding building blocks. Thimble provides the basic code, so students can start editing within the <p> </p> tags. Each student will need to create an account so they can save their work, and for this they will need an email address, so you will need to arrange to give them one if they don't already have one. In Notepad, students will have to save the file manually with a .htm extension. A list of the basic elements of HTML coding can be found in the HTML glossary and the CSS glossary, but please note that when coding HTML and CSS the American spelling of color is used. You will need to be familiar with the major elements available in HTML, which students might use in their designs, and these can be found at: www.w3.org/community/webed/wiki/HTML/Elements. To display or edit raw HTML files, right click the file and open it using a text editor such as Microsoft Notepad or Notepad Plus for PCs, or EDITRA for Mac. If you simply double click the file you will open the HTML document in a web browser.	Communication and networks	3 weeks @ 45mins to 1 hour per week.

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 8: Designing for HCI: a handheld digital device	This unit introduces students to the concept and principles of Human–Computer Interaction and its importance in providing usable solutions for a range of audiences and needs.	Students are challenged to design, for a specific user group and using future technology, a hand-held digital device to include phone functionality.	You will be helping students appreciate why Human–Computer Interaction (HCI) is important when designing systems for human users. A definition of HCI is: Human–Computer Interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them. *Association for Computing Machinery* The design and development of, for example, displays, alarms and interfaces for screens of all sizes, some of which might involve interaction, involves an understanding of the relationship between humans and computers, and it is precisely this relationship that the study of HCI is concerned with. The design and development process needs to be guided by insights from psychology and informed by an appreciation of human diversity. How does your interaction with a computer screen change if you are colour-blind or deaf, for example, and how can the designer mitigate any problems you might encounter, perhaps even enhancing your experience beyond your expectations? It is also important to note that in certain situations there are crucial reliability standards and other performance and safety requirements that have to be met and which might impact on the design, and so should be considered in the design process.	Information technology Hardware and processing Data and representation	3 weeks @ 45mins to 1 hour per week.

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 9: Designing for HCI: an operating system interface	Building on Unit 8, this unit covers the importance of operating systems and how they might develop in relation to on-going changes in technology.	Students are challenged to design an interface for the operating system of the hand-held digital device that they designed in Unit 8. Students will need to think about the technology that will be available in the future and about the needs of their specific user group, to help them with their designs.	The Unit focuses on a range of operating systems and you should be familiar with these and their features. You will need to know how to describe what an operating system does and be aware of current and possible future trends in this area. BBC Click (www.bbc.co.uk/programmes/n13xtmd5) can be a valuable source of information.	Hardware and processing Information technology	3 weeks @ 45mins to 1 hour per week.

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 10: Representing images	This unit is designed to provide students with an understanding of how images and the colours within them are constructed in terms of binary and pixels. Resolution and image types are covered. Stenography – hiding messages in text and images – is covered. Followed by a lesson that requires students to look at how sequenced images can be used to create video (moving images).	Students are challenged to learn about static images so they can stream a video.	You'll be asking students to create one-bit (two colour options, 0 or 1) images to demonstrate how binary and pixels are used to display images on a computer screen. We recommend using spreadsheet software to complete this task. Students will also learn how colour depth affects the look of an image and how, by using binary, we understand how many colours are available to us. We recommend using BMP files and free software called IrfanView (www.irfanview.com) because we can guarantee that nothing unexpected will happen to the file sizes when changing the colour depth. If you choose to use different software, then you will need to check this activity prior to teaching it because the compression techniques used with JPEGs might increase the file size. To change the colour depth in IrfanView, open the image and click on 'Image' in the toolbar, then click 'Reduce colour depth' and enter the number of colours required. In the final lesson, students focus on digital steganography – hiding data within computer files. It is especially important to understand how Least Significant Bit steganography works. It is the process of changing the binary bit that has the smallest impact on the file in which the data is being hidden. In an eight-bit string, this is the bit furthest to the right. The second main activity in the lesson makes use of a number of other cipher techniques. It is worthwhile looking over H.L. Dennis' Secret Breakers website because it covers all of the techniques used: http://hldennis.com/team-veritas/other-secrets-to-break.	Data and data representation	3 weeks @ 45mins to 1 hour per week.

Unit name	Topic	Challenge	Overview of content	Progression Pathway strands covered	Teaching time
Unit 11: Programming a calculator	This unit covers the use of different languages to program and execute a calculator for use by primary school students to solve defined problems.	Students are presented with the scenario whereby children in a primary school have asked them to program a simple shape calculator (using a programming language of the students' choice such as Scratch) to help them with their maths. We have provided students with a page from the primary Maths textbook to give students with an idea of the type of calculations they need help with.	Students are introduced to the principles of algebraic notation and what program variables are used for. They should – by the end of the three lessons in the unit be able to convert simple calculations into standard algebraic notation to model a simple calculator and be able to create a simple calculator program based on a model. Some students could also use standard notation to model a simple calculator with a loop, use good naming conventions of variables and create a program based on the model using a text-based programming language. In the second lesson all students must understand how a simple 'if' selection works and be able to plan a process where a choice of operation can be made. The third lesson is all about procedures and functions. Both are explained in the Student's Book, but you can find more detailed definitions here: www.webopedia.com/TERM/F/function.html and here: http://wiki.scratch.mit.edu/wiki/Procedures. Again, we have supplied resources based on Version 2.0 of Scratch, but you can use other programming languages. Although we have provided you with an off-the-shelf program to use, it helps students if you are able to demonstrate how to program the example used in the Student's Book. The 'Unit 11 Scratch tutorial for teachers' walks you through the more tricky bits and it is advisable to spend some time familiarising yourself with the process before the lesson. The final lesson also refers to subroutines. These are blocks of code that can be called by the main program and executed. We have already met procedures, which are a form of subroutine, and procedures and functions are explored further here.	Programming and development Algorithms	3 weeks @ 45mins to 1 hour per week.

Unit name	**Topic**	**Challenge**	**Overview of content**	**Progression Pathway strands covered**	**Teaching time**
Unit 12: Programming a quiz	Developing further the principles and skills developed in Unit 11, this extends students' work to program a quiz.	The challenge for this unit is to program a maths quiz for primary school pupils. The quiz should ask the player for their name, and then use this in the questions. The quiz should contain sections of questions, each covering a different maths topic. At least one section should contain questions that the computer has generated randomly. All the answers should be numerical.	In the first lesson students will abstract the stages of a simple quiz and write a simple quiz program with score variables. The activities also cover decomposing a quiz program; matching the abstraction steps to a graphical programming language; creating a variable that records a user's name and using it within the quiz; debugging a simple graphical programming problem and using a range of variable types. The second lesson goes into more depth into the subjects of decomposition and generalisation, and looks at random number generators. The use of repeat and make blocks in Scratch are important in this lesson and you should be familiar with how they work. At the end of the lesson you will also cover how to target a quiz design to the specific end user – in this case primary school children. In the lesson objectives for the final lesson the term 'event-driven programming' is used. You need to be familiar with this concept. It describes a program where the flow of the program is driven by events. For example, in Scratch, one block broadcasts or issues a message to another block and the second block runs when it receives the message. This last lesson in the unit covers how to add a timer to a quiz program, and makes the distinction between a count-up timer and a count-down timer. We have supplied resources based on Version 2.0 of Scratch but you can use a range of other graphical programming languages such as BYOB, LEGO Mindstorms, Alice, App Inventor or App Shed.	Algorithms Programming and development	3 weeks @ 45mins to 1 hour per week.