



QuickStart Computing

A CPD toolkit for secondary teachers

FUNDED BY



Department
for Education



Microsoft

Contents

The new National Curriculum for Computing came into effect in September 2014. It presents many exciting opportunities but also some challenges. The challenges are not, however, insurmountable. Regardless of your prior knowledge and experience, this toolkit will help you deliver effective CPD to ensure that you and your team are fully prepared for a creative and innovative curriculum underpinned by computational thinking.

Look at the *User guide*, on page 4, to get started.

Section 1: Leading effective CPD

Pages 6–7

This section helps you envisage effective CPD sessions. You can skip it if you are using this toolkit for self-study.

Section 2: Getting started with confidence

Pages 8–18

This section provides you with all the background information you will need to work out where you are, start planning your new curriculum and teach it with confidence. It unpacks the 2014 National Curriculum Programmes of Study for Computing, explaining the key concepts and dispelling common misconceptions. It introduces you to tools you can use to frame your schemes of work (which are also known as ‘units of work’) and consider progression.

It also provides you with an opportunity to audit your confidence in, and knowledge of, computing.

Section 3: A road map for managing change

Pages 19–27

This section walks you through the process of creating new schemes of work, helping you to identify what can be reused from your existing schemes of work and pointing you in the direction of sources of inspiration to fill the gaps.

Section 5: Resources

Pages 35–52

This section helps you identify what makes an effective computing activity and points you in the direction of lots of activities that you can borrow and make your own.

Section 6: Assessment and progression

Pages 53–60

This section helps you look beyond the demise of government-prescribed levels and consider how you can assess progression and achievement across the subject of computing.

Section 4: Teaching

Pages 28–34

This section will help you visualise a good computing lesson and provide you with the tools to start planning your own creative and innovative lessons.

Section 7: Next steps

Pages 61–3

This section describes sources of support and opportunities for further CPD.

Foreword

September 2014 saw the introduction of the new National Curriculum in England and, for the first time, schools must teach a new subject – computing.

For most of us technology and computers play a vital role in our lives, at home, at work, for our health and for our informal learning. It is important that our children learn how this stuff works, rather than treating it as magic. Just as we teach the traditional sciences in primary school and into secondary school, we must now teach *computer science*, the ‘fourth

science’, to ensure children leave school equipped with the skills and knowledge they need to participate effectively in society, whether or not they go on to become computing professionals.

The computing curriculum is a once-in-a-lifetime opportunity for schools to make a difference to children’s futures. Let’s make the most of it! I hope that **QuickStart Computing** will help you design, develop and deliver an invigorating computing curriculum that will inspire you and your students.

Simon Peyton Jones
Chair, Computing At School

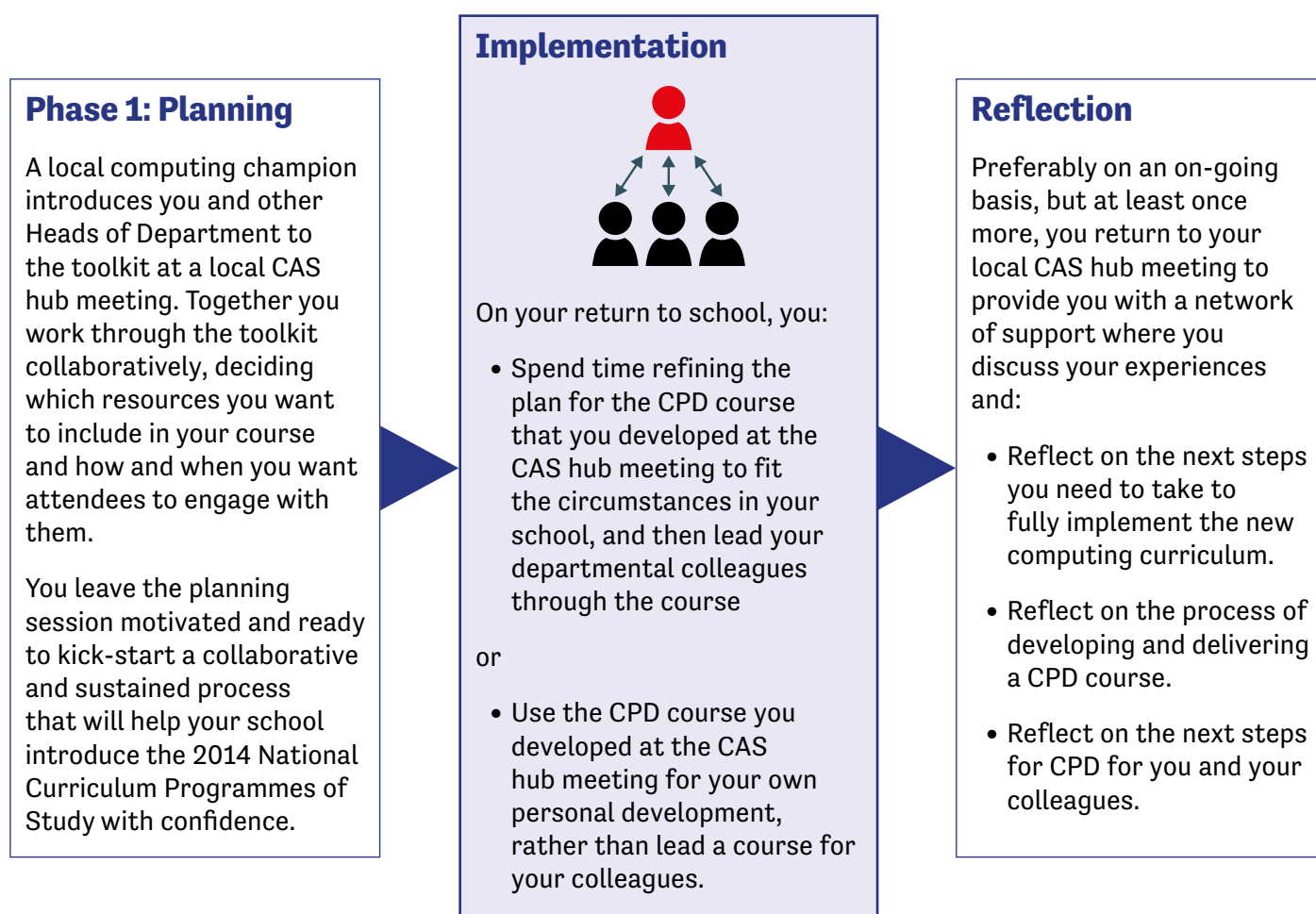
User guide

What will this toolkit help me do?

This pack is not a CPD course in a box. It is a toolkit, and inside it you'll find all the tools you will need to develop a course that helps you and the teachers you work with figure out how to teach a creative and innovative computing curriculum underpinned by computational thinking. It is not about learning specific computing skills. It will not, for example, teach you how to program in Python. What it will do is help you work out for yourself how to manage the transition from the old curriculum to the new curriculum, develop high quality schemes of work, teach high quality computing lessons and decide how you will assess, record and report progression in computing.

You will need to spend time planning your CPD course carefully, selecting which resources are most appropriate to support the journey you, your department and your school are embarking on, and deciding how best to use them. We thoroughly recommend you go to a CAS hub and work with colleagues in a similar position to plan your course collaboratively, before refining the plan to fit your circumstances.

Your journey could look like this:



If you are running a CPD course for the first time, we strongly recommend that you start by reading **Section 1: Leading effective CPD**.

How do I use this toolkit?

The toolkit has been divided into seven sections. Each section begins with an introduction that poses a number of questions, then goes on to explain which resources you can use to answer these questions, how we suggest the resources are used, and – in a table – whether the resources can be found in the book, on the CD-ROM or on the website.



When you see this icon it means that the resources are designed to be used in face-to-face sessions with colleagues.



When you see this icon it means you will get the most out of these resources if you work on them alone and share your findings with colleagues.

For example:

What are the Progression Pathways?

The *What are the Progression Pathways?* information sheet explains why we have presented the Progression Pathways in terms of topics and strands and describes how they can be used to help you assess students' progress in computing. Both versions of the Progression Pathways are provided as information sheets for you to use.

There is an information sheet called *What are the Progression Pathways*, which we suggest you read. It can be found on pages 12–13 of the book, on the CD-ROM and on the website.

		Book	CD-ROM	Website
	Information sheet: <i>What are the Progression Pathways?</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.	✓ Pages 12–13	✓	✓
	Information sheet: <i>Progression Pathways: topics</i> © 2014 Mark Dorling and Matthew Walker. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.	✓ Pages 14–15	✓	✓
	Information sheet: <i>Progression Pathways: strands</i> © 2014 Mark Dorling and Matthew Walker. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.	✓ Pages 16–17	✓	✓

Depending on your level of experience you may find some resources more useful than others.

Section 1: Leading effective CPD

What does high quality CPD look like?

Read *High quality CPD*.

	Book	CD-ROM	Website
Information sheet: <i>High quality CPD</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.	✓ Page 7	✓	✓



High quality CPD

Good CPD and bad CPD

At recent Computing At School (CAS) Master Teacher training sessions, teachers were asked what they thought good and bad CPD looked like.

GOOD CPD

Presentation: *interactive, hands-on, engaging, not too much talking from the front*

Content: *relevant to the curriculum, directly applicable, something to take away*

Trainer: *good subject knowledge, approachable, classroom experience, enthusiastic*

BAD CPD

A dull and unenthusiastic presenter

A presenter who lacks appropriate subject knowledge

Little or no interaction throughout the session

Few resources provided for the session or follow-up

These points will help you start planning a CPD training course. Good CPD sessions need careful planning. However the CAS view is that CPD does not start and end at the actual training event. It is a long-term collaborative process sustained over time. This is backed up by research.

What does the research say?

There are numerous studies relating to how to deliver effective CPD. One meta-study developed by the Centre for the Use of Research Evidence in Education (CUREE) looked at numerous reviews of studies and published a report (CUREE, 2012) describing five criteria that CPD should meet to have the maximum impact on learning. It should be:

1. Collaborative
2. Supported by specialist expertise
3. Focused on aspirations for students
4. Sustained over time
5. Exploring evidence from trying new things

Collaborative CPD involves teachers working together on a sustained basis and/or teachers working with other professional colleagues. In an earlier study by Cordingley et al (2005) looking at 266 studies of collaborative learning, all but one of them was found to support teacher improvement.

So encouraging teachers to work together will facilitate greater results in the classroom.

Being 'sustained over time' means that the CPD needs to last well after the session has finished. This can be achieved by follow-up activities or group tasks, or by making arrangements to check back on progress made. It also means that teachers should be prepared in advance for attending a workshop, perhaps by carrying out an activity beforehand or articulating what they hope to achieve by attending.

In order for CPD to be sustained, it also needs to be relevant to teachers' current teaching as that will give them an opportunity to practise. Another meta-study, by Guskey and Yoon (2009), highlighted that follow-up activities are extremely important, and that CPD workshops should be seen as just one part of a broader programme of professional development activities. This is the approach taken by Aileen Kennedy (2005), who talks about CPD being 'transformative' if it combines a range of CPD approaches which complement each other. Other elements of CPD could include peer coaching, action research projects, and working together in a 'community of practice'. This is very much the CAS approach. Another element of transformative CPD is the opportunity for it to be accredited. One place this is available is via the BCS Certificate in Computer Science Teaching (computingatschool.org.uk/certificate). Attending CPD sessions that you run in school can contribute to the evidence a teacher has to put together to gain this certification.

In summary, the available research enables us to see that as well as planning CPD events that are hands-on, fun, interactive and run by an enthusiastic trainer, we should ensure that the learning from the event takes place before and after the session and that there is an opportunity to work together with colleagues. This will maximise the benefit afforded by the CPD.


References:

- Cordingley, P., Bell, M., Thomason, S. and Firth, A. 'The impact of collaborative continuing professional development (CPD) on classroom teaching and learning. Review: How do collaborative and sustained CPD and sustained but not collaborative CPD affect teaching and learning?', *Research Evidence in Education Library*. London: EPPI-Centre, Social Science Research Unit, Institute of Education, University of London, 2005.
- CUREE. *Understanding What Enables High Quality Professional Learning*, Technical Report (2012).
- Kennedy, A. 'Models of Continuing Professional Development: a framework for analysis', *Journal of In-Service Education*, Vol. 31, No. 2, 2005.
- Guskey, T. and Yoon, K. 'What Works in Professional Development?', *Phi Delta Kappan*, Vol. 90, No. 7, 2009.

Section 2: Getting started with confidence

Where do I start?

Begin by reflecting on what the changes to the National Curriculum mean to you, using the *Time to reflect* planning sheet.

		Book	CD-ROM	Website
	Planning sheet: <i>Time to reflect</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.	✓ Page 11	✓	✓

I have so many questions. Where do I find the answers?



You are probably asking yourself:






- How has the National Curriculum changed?
- When do I need to start teaching the 2014 National Curriculum Programmes of Study for Computing?
- Why is the inclusion of Computer Science important?
- How do Computer Science, Information Technology and Digital Literacy differ and fit together?
- What does the terminology used in the National Curriculum Programmes of Study for Computing mean?

Watch the *Introducing the 2014 National Curriculum* video and Simon Peyton Jones' inspirational TEDxExeter talk, the *Teaching creative computer science* video, as a way in to answering these fundamental questions. If you haven't already, take a good look at the *National Curriculum Programmes of Study for Computing*, which are provided as information sheets.

In the *Introducing the 2014 National Curriculum* video, Mark Dorling mentions the Computing At School primary and secondary guidance documents. *Computing in the National Curriculum: a guide for primary teachers* explains where students are coming from and *Computing in the National Curriculum: a guide for secondary teachers* explains where they are going. Both guides, which are provided as information sheets, are a useful starting point if you are struggling to understand the new terminology being used. *QuickStart Computing: A CPD toolkit for primary teachers* also contains valuable subject knowledge, which you may find useful, and a link is provided.


Return to the *Time to reflect* planning sheet and complete it again, this time in a different colour. Hopefully you'll be able to see that your confidence has developed in a very short period of time.

		Book	CD-ROM	Website
	Video: <i>Introducing the 2014 National Curriculum</i> © Hodder Education 2014. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.		✓	✓
	Video: <i>Teaching creative computer science</i> © TEDxExeter 2014. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.		✓	✓

		Book	CD-ROM	Website
	Information sheet: <i>National Curriculum in England: computing programmes of study – Key Stages 1 and 2</i> © Crown Copyright 2013. Content is available under the Open Government Licence v2.0.		✓	✓
	Information sheet: <i>National Curriculum in England: computing programmes of study – Key Stages 3 and 4</i> © Crown Copyright 2013. Content is available under the Open Government Licence v2.0.		✓	✓
	Information sheet: <i>Computing in the National Curriculum: a guide for primary teachers</i> © Computing At School 2013. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.		✓	✓
	Information sheet: <i>Computing in the National Curriculum: a guide for secondary teachers</i> © Computing At School 2013. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.		✓	✓
	Link: QuickStart Computing: A CPD toolkit for primary teachers		✓	✓



What are the Progression Pathways?

The *What are the Progression Pathways?* information sheet explains why we have presented the Progression Pathways in terms of topics and strands and describes how they can be used to help you assess students' progress in computing. Both versions of the Progression Pathways are provided as information sheets for you to use.

		Book	CD-ROM	Website
	Information sheet: <i>What are the Progression Pathways?</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.	✓ Pages 12–13	✓	✓
	Information sheet: <i>Progression Pathways: topics</i> © 2014 Mark Dorling and Matthew Walker. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.	✓ Pages 14–15	✓	✓
	Information sheet: <i>Progression Pathways: strands</i> © 2014 Mark Dorling and Matthew Walker. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.	✓ Pages 16–17	✓	✓


What is the significance of Computational Thinking (CT) and what role does problem solving play in CT?

Watch the *Developing computational thinking in the classroom* video and take a look at *Developing computational thinking in the classroom – a framework*, also known as the 'Computational Thinking Framework', which is provided as an information sheet.

		Book	CD-ROM	Website
	Video: <i>Developing computational thinking in the classroom</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.		✓	✓
	Link: Developing computational thinking in the classroom – a framework		✓	✓

What is the difference between ‘computing’ and ‘coding and programming’?

This is a thorny issue and reading the *Computing ≠ coding and programming* information sheet should help clarify things for you.





		Book	CD-ROM	Website
	Information sheet: <i>Computing ≠ coding and programming</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.	✓ Page 18	✓	✓

How do I audit my skills and knowledge?

Many of you will feel very nervous about teaching computing because you do not have the background subject knowledge and skills. This is absolutely to be expected, and isn't something you should be embarrassed about or feel forced to hide. Great teaching is all about being reflective and it is important to share your experiences, both good and bad. Computing is a relatively new subject in comparison to subjects such as mathematics or physics. It is being taught at Key Stages 1, 2 and 3 for the very first time from September 2014, and it is quite likely that some aspects of the subject are being taught in your school for the first time. To tackle this problem head on, complete the interactive *Skills and knowledge audit* to identify your areas of weakness. You can use the resulting information to discuss your training needs with your Head of Department. Heads of Department can collate the information collected confidentially from each member of their department and present the Senior Leadership Team with detailed CPD requirements.

The links to the *Gatsby subject knowledge diagnosis tool for computing teachers* also provide two tests, one for primary school teachers and one for secondary school teachers, to indicate gaps in subject knowledge in the following five areas: algorithms, programming, data, computers and social informatics, communication and the Internet.

Both the *Skills and knowledge audit* and the *Gatsby subject knowledge diagnosis tool for computing teachers* are based on the *Subject knowledge requirements for entry into computer science teacher training* produced by the Teaching Agency, which is provided as an information sheet.

		Book	CD-ROM	Website
	Interactive: <i>Skills and knowledge audit</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.		✓	✓
	Link: <i>Gatsby subject knowledge diagnosis tool for computing teachers – primary</i>		✓	✓
	Link: <i>Gatsby subject knowledge diagnosis tool for computing teachers – secondary</i>		✓	✓
	Information sheet: <i>Subject knowledge requirements for entry into computer science teacher training</i> © Crown Copyright 2012. Content is available under the Open Government Licence.		✓	✓

Round off this section by preparing a presentation for your Senior Leadership Team summarising the changes to the National Curriculum and what they mean for you, for your department and for your school.

Time to reflect

The National Curriculum has changed.... Take some time to reflect on what this means for you by completing the table below.

What facts do I already know about the changes? What facts do I need to find out about the changes?	
What do I feel about the changes?	
What benefits will come out of the changes?	
Are there any potential problems that will come out of the changes? For example, a lack of specialist subject knowledge.	
Do the changes present any new and exciting opportunities for me to do things differently?	
Are there any challenges that will arise if I do things differently?	

What are the Progression Pathways?

Why are there two versions of the Progression Pathways?

The two versions of the Progression Pathways are as follows:

1. Progression Pathways: topics

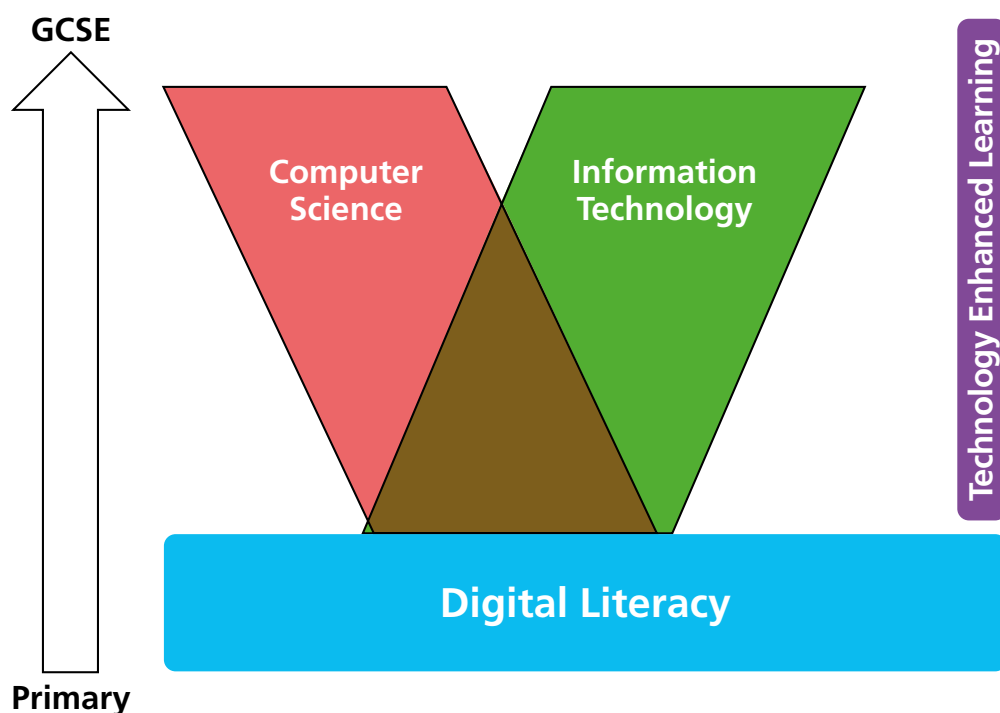
Progression Pathways: topics presents the subject of computing as a whole, to encourage you and your students to think about a broad, balanced and integrated curriculum that includes computer science, information technology and digital literacy. It is not until Key Stage 4, when students can choose between qualifications in computer science or qualifications in information technology, that they should become aware of the different strands that exist within computing. Throughout Key Stages 1, 2 and 3 the focus should be on integrating these strands and developing students' computational thinking skills.

Algorithms	Programming & Development	Data & Data Representation
<ul style="list-style-type: none"> Understands what an algorithm is and is able to express simple linear (non-branching) algorithms symbolically. (AL) Understands that computers need precise instructions. (AL) Demonstrates care and precision to avoid errors. (AL) 	<ul style="list-style-type: none"> Knows that users can develop their own programs, and can demonstrate this by creating a simple program in an environment that does not rely on text e.g. programmable robots etc. (AL) Executes, checks and changes programs. (AL) Understands that programs execute by following precise instructions. (AL) 	<ul style="list-style-type: none"> Recognises that digital content can be represented in many forms. (AB) (GE) Distinguishes between some of these forms and can explain the different ways that they communicate information. (AB)
Hardware & Processing	Communication & Networks	Information Technology
<ul style="list-style-type: none"> Understands that computers have no intelligence and that computers can do nothing unless a program is executed. (AL) Recognises that all software executed on digital devices is programmed. (AL) (AB) (GE) 	<ul style="list-style-type: none"> Obtains content from the world wide web using a web browser. (AL) Understands the importance of communicating safely and respectfully online, and the need for keeping personal information private. (EV) Knows what to do when concerned about content or being contacted. (AL) 	<ul style="list-style-type: none"> Uses software under the control of the teacher to create, store and edit digital content using appropriate file and folder names. (AB) (GE) (DE) Understands that people interact with computers. Shares their use of technology in school. Knows common uses of information technology beyond the classroom. (GE) Talks about their work and makes changes to improve it. (EV)

2. Progression Pathways: strands

Progression Pathways: strands does break the subject of computing into the three strands of computer science, information technology and digital literacy to illustrate why you do not have to throw out your current curriculum and start entirely from scratch. IT still plays an important role in the curriculum, and digital literacy has an expanded role. This version of the Progression Pathways is designed to empower you to audit your existing curriculum and fill in the gaps.

Computer Science	Information Technology	Digital Literacy
<p>Understands what an algorithm is and is able to express simple linear (non-branching) algorithms symbolically. Understands that computers need precise instructions. Demonstrates care and precision to avoid errors. (AL)</p> <p>Knows that users can develop their own programs, and can demonstrate this by creating a simple program in an environment that does not rely on text e.g. programmable robots etc. Executes, checks and changes programs. Understands that programs execute by following precise instructions. (AL)</p> <p>Understands that computers have no intelligence and that computers can do nothing unless a program is executed. (AL)</p> <p>Recognises that all software executed on digital devices is programmed. (AL) (AB) (GE)</p>	<p>Recognises that digital content can be represented in many forms. (AB) (GE) Distinguishes between some of these forms and can explain the different ways that they communicate information. (AB)</p> <p>Obtains content from the world wide web using a web browser. (AL)</p> <p>Uses software under the control of the teacher to create, store and edit digital content using appropriate file and folder names. (AB) (GE) (DE) Understands that people interact with computers.</p> <p>Talks about their work and makes changes to improve it. (EV)</p>	<p>Understands the importance of communicating safely and respectfully online, and the need for keeping personal information private. (EV)</p> <p>Knows what to do when concerned about content or being contacted. (AL)</p> <p>Knows common uses of information technology beyond the classroom. (GE)</p> <p>Shares their use of technology in school.</p>



A diagram showing how the strands of computing relate to each other over the key stages.

What are the Progression Pathways for?

The Progression Pathways are based on the 2014 National Curriculum Computing Programmes of Study and are designed to help you assess students' progress in computing. As students demonstrate competence in the statements in each colour-coded row, moving from the top row to the bottom row of the Pathways, you can recognise achievement and/or attainment. Arbitrary values (which could be referred to as 'levels') could be assigned to each colour-coded row to enable you to use your assessment findings within an existing school-wide reporting system.


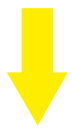






It is suggested that the pink, yellow, orange, blue and some of the purple rows are the most appropriate for Key Stages 1 and 2; and some of the purple, red and black rows are the most appropriate for Key Stage 3. The white row overlaps with current Level 2 qualifications. However, students currently in Key Stage 3 or entering Key Stage 3 over the next year or so may not have the prior learning required to begin their learning at the purple row, because the National Curriculum Programme of Study for Computing is being introduced into Key Stages 1, 2 and 3 simultaneously. It is therefore important to ascertain where your students are on the Pathways before you begin teaching and assign arbitrary values accordingly.

Rewarding achievement and attainment

Digital badges, named after the column headings in the version of the Progression Pathways you opt to use and coloured according to the row or 'level' met, could be used to reward achievement and attainment. Two-tone badges could be used to reward students who are working between two colours. And designing the digital badges themselves will give students a greater ownership of the rewards system.



Computing Progression Pathways

Pupil Progression	Algorithms	Programming & Development	Data & Data Representation
	<ul style="list-style-type: none"> Understands what an algorithm is and is able to express simple linear (non-branching) algorithms symbolically. (AL) Understands that computers need precise instructions. (AL) Demonstrates care and precision to avoid errors. (AL) 	<ul style="list-style-type: none"> Knows that users can develop their own programs, and can demonstrate this by creating a simple program in an environment that does not rely on text e.g. programmable robots etc. (AL) Executes, checks and changes programs. (AL) Understands that programs execute by following precise instructions. (AL) 	<ul style="list-style-type: none"> Recognises that digital content can be represented in many forms. (AB) (GE) Distinguishes between some of these forms and can explain the different ways that they communicate information. (AB)
	<ul style="list-style-type: none"> Understands that algorithms are implemented on digital devices as programs. (AL) Designs simple algorithms using loops, and selection i.e. if statements. (AL) Uses logical reasoning to predict outcomes. (AL) Detects and corrects errors i.e. debugging, in algorithms. (AL) 	<ul style="list-style-type: none"> Uses arithmetic operators, if statements, and loops, within programs. (AL) Uses logical reasoning to predict the behaviour of programs. (AL) Detects and corrects simple semantic errors i.e. debugging, in programs. (AL) 	<ul style="list-style-type: none"> Recognises different types of data: text, number. (AB) (GE) Appreciates that programs can work with different types of data. (GE) Recognises that data can be structured in tables to make it useful. (AB) (DE)
	<ul style="list-style-type: none"> Designs solutions (algorithms) that use repetition and two-way selection i.e. if, then and else. (AL) Uses diagrams to express solutions. (AB) Uses logical reasoning to predict outputs, showing an awareness of inputs. (AL) 	<ul style="list-style-type: none"> Creates programs that implement algorithms to achieve given goals. (AL) Declares and assigns variables. (AB) Uses post-tested loop e.g. 'until', and a sequence of selection statements in programs, including an if, then and else statement. (AL) 	<ul style="list-style-type: none"> Understands the difference between data and information. (AB) Knows why sorting data in a flat file can improve searching for information. (EV) Uses filters or can perform single criteria searches for information. (AL)
	<ul style="list-style-type: none"> Shows an awareness of tasks best completed by humans or computers. (EV) Designs solutions by decomposing a problem and creates a sub-solution for each of these parts. (DE) (AL) (AB) Recognises that different solutions exist for the same problem. (AL) (AB) 	<ul style="list-style-type: none"> Understands the difference between, and appropriately uses if and if, then and else statements. (AL) Uses a variable and relational operators within a loop to govern termination. (AL) (GE) Designs, writes and debugs modular programs using procedures. (AL) (DE) (AB) (GE) Knows that a procedure can be used to hide the detail with sub-solution. (AL) (DE) (AB) (GE) 	<ul style="list-style-type: none"> Performs more complex searches for information e.g. using Boolean and relational operators. (AL) (GE) (EV) Analyses and evaluates data and information, and recognises that poor quality data leads to unreliable results, and inaccurate conclusions. (AL) (EV)
	<ul style="list-style-type: none"> Understands that iteration is the repetition of a process such as a loop. (AL) Recognises that different algorithms exist for the same problem. (AL) (GE) Represents solutions using a structured notation. (AL) (AB) Can identify similarities and differences in situations and can use these to solve problems (pattern recognition). (GE) 	<ul style="list-style-type: none"> Understands that programming bridges the gap between algorithmic solutions and computers. (AB) Has practical experience of a high-level textual language, including using standard libraries when programming. (AB) (AL) Uses a range of operators and expressions e.g. Boolean, and applies them in the context of program control. (AL) Selects the appropriate data types. (AL) (AB) 	<ul style="list-style-type: none"> Knows that digital computers use binary to represent all data. (AB) Understands how bit patterns represent numbers and images. (AB) Knows that computers transfer data in binary. (AB) Understands the relationship between binary and file size (uncompressed). (AB) Defines data types: real numbers and Boolean. (AB) Queries data on one table using a typical query language. (AB)
	<ul style="list-style-type: none"> Understands a recursive solution to a problem repeatedly applies the same solution to smaller instances of the problem. (AL) (GE) Recognises that some problems share the same characteristics and use the same algorithm to solve both. (AL) (GE) Understands the notion of performance for algorithms and appreciates that some algorithms have different performance characteristics for the same task. (AL) (EV) 	<ul style="list-style-type: none"> Uses nested selection statements. (AL) Appreciates the need for, and writes, custom functions including use of parameters. (AL) (AB) Knows the difference between, and uses appropriately, procedures and functions. (AL) (AB) Understands and uses negation with operators. (AL) Uses and manipulates one dimensional data structures. (AB) Detects and corrects syntactical errors. (AL) 	<ul style="list-style-type: none"> Understands how numbers, images, sounds and character sets use the same bit patterns. (AB) (GE) Performs simple operations using bit patterns e.g. binary addition. (AB) (AL) Understands the relationship between resolution and colour depth, including the effect on file size. (AB) Distinguishes between data used in a simple program (a variable) and the storage structure for that data. (AB)
	<ul style="list-style-type: none"> Recognises that the design of an algorithm is distinct from its expression in a programming language (which will depend on the programming constructs available). (AL) (AB) Evaluates the effectiveness of algorithms and models for similar problems. (AL) (AB) (GE) Recognises where information can be filtered out in generalizing problem solutions. (AL) (AB) (GE) Uses logical reasoning to explain how an algorithm works. (AL) (AB) (DE) Represents algorithms using structured language. (AL) (DE) (AB) 	<ul style="list-style-type: none"> Appreciates the effect of the scope of a variable e.g. a local variable can't be accessed from outside its function. (AB) (AL) Understands and applies parameter passing. (AB) (GE) (DE) Understands the difference between, and uses, both pre-tested e.g. 'while', and post-tested e.g. 'until' loops. (AL) Applies a modular approach to error detection and correction. (AB) (DE) (GE) 	<ul style="list-style-type: none"> Knows the relationship between data representation and data quality. (AB) Understands the relationship between binary and electrical circuits, including Boolean logic. (AB) Understands how and why values are data typed in many different languages when manipulated within programs. (AB)
	<ul style="list-style-type: none"> Designs a solution to a problem that depends on solutions to smaller instances of the same problem (recursion). (AL) (DE) (AB) (GE) Understands that some problems cannot be solved computationally. (AB) (GE) 	<ul style="list-style-type: none"> Designs and writes nested modular programs that enforce reusability utilising sub-routines wherever possible. (AL) (AB) (GE) (DE) Understands the difference between 'While' loop and 'For' loop, which uses a loop counter. (AL) (AB) Understands and uses two dimensional data structures. (AB) (DE) 	<ul style="list-style-type: none"> Performs operations using bit patterns e.g. conversion between binary and hexadecimal, binary subtraction etc. (AB) (AL) (GE) Understands and can explain the need for data compression, and performs simple compression methods. (AL) (AB) Knows what a relational database is, and understands the benefits of storing data in multiple tables. (AB) (GE) (DE)

Computational Thinking Concept: AB = Abstraction; DE = Decomposition; AL = Algorithmic Thinking; EV = Evaluation; GE = Generalisation

Note: Each of the Progression Pathway statements is underpinned by one-or-more learning outcomes (due for publication in 2014), providing greater detail of what should be taught to achieve each Progression Pathway statement and National Curriculum point of study.

Hardware & Processing	Communication & Networks	Information Technology
<ul style="list-style-type: none"> Understands that computers have no intelligence and that computers can do nothing unless a program is executed. (AL) Recognises that all software executed on digital devices is programmed. (AL) (AB) (GE) 	<ul style="list-style-type: none"> Obtains content from the world wide web using a web browser. (AL) Understands the importance of communicating safely and respectfully online, and the need for keeping personal information private. (EV) Knows what to do when concerned about content or being contacted. (AL) 	<ul style="list-style-type: none"> Uses software under the control of the teacher to create, store and edit digital content using appropriate file and folder names. (AB) (GE) (DE) Understands that people interact with computers. Shares their use of technology in school. Knows common uses of information technology beyond the classroom. (GE) Talks about their work and makes changes to improve it. (EV)
<ul style="list-style-type: none"> Recognises that a range of digital devices can be considered a computer. (AB) (GE) Recognises and can use a range of input and output devices. Understands how programs specify the function of a general purpose computer. (AB) 	<ul style="list-style-type: none"> Navigates the web and can carry out simple web searches to collect digital content. (AL) (EV) Demonstrates use of computers safely and responsibly, knowing a range of ways to report unacceptable content and contact when online. 	<ul style="list-style-type: none"> Uses technology with increasing independence to purposefully organise digital content. (AB) Shows an awareness for the quality of digital content collected. (EV) Uses a variety of software to manipulate and present digital content: data and information. (AL) Shares their experiences of technology in school and beyond the classroom. (GE) (EV) Talks about their work and makes improvements to solutions based on feedback received. (EV)
<ul style="list-style-type: none"> Knows that computers collect data from various input devices, including sensors and application software. (AB) Understands the difference between hardware and application software, and their roles within a computer system. (AB) 	<ul style="list-style-type: none"> Understands the difference between the internet and internet service e.g. world wide web. (AB) Shows an awareness of, and can use a range of internet services e.g. VOIP. Recognises what is acceptable and unacceptable behaviour when using technologies and online services. 	<ul style="list-style-type: none"> Collects, organises and presents data and information in digital content. (AB) Creates digital content to achieve a given goal through combining software packages and internet services to communicate with a wider audience e.g. blogging. (AL) Makes appropriate improvements to solutions based on feedback received, and can comment on the success of the solution. (EV)
<ul style="list-style-type: none"> Understands why and when computers are used. (EV) Understands the main functions of the operating system. (DE) (AB) Knows the difference between physical, wireless and mobile networks. (AB) 	<ul style="list-style-type: none"> Understands how to effectively use search engines, and knows how search results are selected, including that search engines use 'web crawler programs'. (AB) (GE) (EV) Selects, combines and uses internet services. (EV) Demonstrates responsible use of technologies and online services, and knows a range of ways to report concerns. 	<ul style="list-style-type: none"> Makes judgements about digital content when evaluating and repurposing it for a given audience. (EV) (GE) Recognises the audience when designing and creating digital content. (EV) Understands the potential of information technology for collaboration when computers are networked. (GE) Uses criteria to evaluate the quality of solutions, can identify improvements making some refinements to the solution, and future solutions. (EV)
<ul style="list-style-type: none"> Recognises and understands the function of the main internal parts of basic computer architecture. (AB) Understands the concepts behind the fetch-execute cycle. (AB) (AL) Knows that there is a range of operating systems and application software for the same hardware. (AB) 	<ul style="list-style-type: none"> Understands how search engines rank search results. (AL) Understands how to construct static web pages using HTML and CSS. (AL) (AB) Understands data transmission between digital computers over networks, including the internet i.e. IP addresses and packet switching. (AL) (AB) 	<ul style="list-style-type: none"> Evaluates the appropriateness of digital devices, internet services and application software to achieve given goals. (EV) Recognises ethical issues surrounding the application of information technology beyond school. Designs criteria to critically evaluate the quality of solutions, uses the criteria to identify improvements and can make appropriate refinements to the solution. (EV)
<ul style="list-style-type: none"> Understands the von Neumann architecture in relation to the fetch-execute cycle, including how data is stored in memory. (AB) (GE) Understands the basic function and operation of location addressable memory. (AB) 	<ul style="list-style-type: none"> Knows the names of hardware e.g. hubs, routers, switches, and the names of protocols e.g. SMTP, iMAP, POP, FTP, TCP/IP, associated with networking computer systems. (AB) Uses technologies and online services securely, and knows how to identify and report inappropriate conduct. (AL) 	<ul style="list-style-type: none"> Justifies the choice of and independently combines and uses multiple digital devices, internet services and application software to achieve given goals. (EV) Evaluates the trustworthiness of digital content and considers the usability of visual design features when designing and creating digital artifacts for a known audience. (EV) Identifies and explains how the use of technology can impact on society. Designs criteria for users to evaluate the quality of solutions, uses the feedback from the users to identify improvements and can make appropriate refinements to the solution. (EV)
<ul style="list-style-type: none"> Knows that processors have instruction sets and that these relate to low-level instructions carried out by a computer. (AB) (AL) (GE) 	<ul style="list-style-type: none"> Knows the purpose of the hardware and protocols associated with networking computer systems. (AB) (AL) Understands the client-server model including how dynamic web pages use server-side scripting and that web servers process and store data entered by users. (AL) (AB) (DE) Recognises that persistence of data on the internet requires careful protection of online identity and privacy. 	<ul style="list-style-type: none"> Undertakes creative projects that collect, analyse, and evaluate data to meet the needs of a known user group. (AL) (DE) (EV) Effectively designs and creates digital artefacts for a wider or remote audience. (AL) (DE) Considers the properties of media when importing them into digital artifacts. (AB) Documents user feedback, the improvements identified and the refinements made to the solution. (AB) Explains and justifies how the use of technology impacts on society, from the perspective of social, economical, political, legal, ethical and moral issues. (EV)
<ul style="list-style-type: none"> Has practical experience of a small (hypothetical) low level programming language. (AB) (AL) (DE) (GE) Understands and can explain Moore's Law. (GE) Understands and can explain multitasking by computers. (AB) (AL) (DE) 	<ul style="list-style-type: none"> Understands the hardware associated with networking computer systems, including WANs and LANs, understands their purpose and how they work, including MAC addresses. (AB) (AL) (DE) (GE) 	<ul style="list-style-type: none"> Understands the ethical issues surrounding the application of information technology, and the existence of legal frameworks governing its use e.g. Data Protection Act, Computer Misuse Act, Copyright etc. (EV)

Computing Progression Pathways –

Mapped to Computer Science, Information Technology and Digital Literacy strands of the National Curriculum Programme of Study

Pupil Progression

Computer Science

Understands what an algorithm is and is able to express simple linear (non-branching) algorithms symbolically. Understands that computers need precise instructions. Demonstrates care and precision to avoid errors. (AL)

Knows that users can develop their own programs, and can demonstrate this by creating a simple program in an environment that does not rely on text e.g. programmable robots etc. Executes, checks and changes programs. Understands that programs execute by following precise instructions. (AL)

Understands that computers have no intelligence and that computers can do nothing unless a program is executed. (AL) (AB) (GE)

Understands that algorithms are implemented on digital devices as programs. Designs simple algorithms using loops, and selection i.e. if statements. Uses logical reasoning to predict outcomes. Detects and corrects errors i.e. debugging, in algorithms. (AL)

Uses arithmetic operators, if statements, and loops, within programs. Uses logical reasoning to predict the behaviour of programs. Detects and corrects simple semantic errors i.e. debugging, in programs. (AL)

Recognises that a range of digital devices can be considered a computer. (AB) (GE) Recognises and can use a range of input and output devices. Understands how programs specify the function of a general purpose computer. (AB)

Designs solutions (algorithms) that use repetition and two-way selection i.e. if, then and else. (AL) Uses diagrams to express solutions. (AB) Uses logical reasoning to predict outputs, showing an awareness of inputs. (AL)

Creates programs that implement algorithms to achieve given goals. (AL) Declares and assigns variables. (AB) Uses post-tested loop e.g. 'until', and a sequence of selection statements in programs, including an if, then and else statement. (AL)

Knows that computers collect data from various input devices, including sensors and application software. (AB) Understands the difference between hardware and application software, and their roles within a computer system. (AB)

Understands the difference between the internet and internet service e.g. world wide web. (AB)

Shows an awareness of tasks best completed by humans or computers. (EV) Designs solutions by decomposing a problem and creates a sub-solution for each of these parts (decomposition). (DE) (AL) (AB) Recognises that different solutions exist for the same problem. (AL) (AB)

Understands the difference between, and appropriately uses if and if, then and else statements. (AL) Uses a variable and relational operators within a loop to govern termination. (AL) (GE) Designs, writes and debugs modular programs using procedures. (AL) (DE) (AB) (GE) Knows that a procedure can be used to hide the detail with sub-solution (procedural abstraction). (AL) (DE) (AB) (GE)

Understands why and when computers are used. (EV) Understands the main functions of the operating system. (DE) (AB) Understands how to effectively use search engines, and knows how search results are selected, including that search engines use 'web crawler programs'. (AB) (GE) (EV)

Understands that iteration is the repetition of a process such as a loop. (AL) Recognises that different algorithms exist for the same problem. (AL) (GE) Represents solutions using a structured notation. (AL) (AB) Can identify similarities and differences in situations and can use these to solve problems (pattern recognition). (GE)

Understands that programming bridges the gap between algorithmic solutions and computers. (AB) Has practical experience of a high-level textual language, including using standard libraries when programming. (AB) (AL) Uses a range of operators and expressions e.g. Boolean, and applies them in the context of program control. (AL) Selects the appropriate data types. (AL) (AB)

Information Technology

Recognises that digital content can be represented in many forms. (AB) (GE) Distinguishes between some of these forms and can explain the different ways that they communicate information. (AB)

Obtains content from the world wide web using a web browser. (AL) Uses software under the control of the teacher to create, store and edit digital content using appropriate file and folder names. (AB) (GE) (DE) Understands that people interact with computers.

Talks about their work and makes changes to improve it. (EV)

Recognises different types of data: text, number. (AB) (GE) Appreciates that programs can work with different types of data. (GE) Recognises that data can be structured in tables to make it useful. (AB) (DE)

Recognises that a range of digital devices can be considered a computer. (AB) (GE) Recognises and can use a range of input and output devices.

Navigates the web and can carry out simple web searches to collect digital content. (AL) (EV)

Uses technology with increasing independence to purposefully organise digital content. (AB)

Uses a variety of software to manipulate and present digital content: data and information. (AL) Shares their experiences of technology in school and beyond the classroom. (GE) (EV) Talks about their work and makes improvements to solutions based on feedback received. (EV)

Understands the difference between data and information. (AB) Knows why sorting data in a flat file can improve searching for information. (EV) Uses filters or can perform single criteria searches for information. (AL)

Shows an awareness of, and can use a range of internet services e.g. VOIP.

Collects, organises and presents data and information in digital content. (AB) Creates digital content to achieve a given goal through combining software packages and internet services to communicate with a wider audience e.g. blogging. (AL) Makes appropriate improvements to solutions based on feedback received, and can comment on the success of the solution. (EV)

Performs more complex searches for information e.g. using Boolean and relational operators. (AL) (GE) (EV) Analyses and evaluates data and information, and recognises that poor quality data leads to unreliable results, and inaccurate conclusions. (AL) (EV)

Knows the difference between physical, wireless and mobile networks. (AB)

Recognises the audience when designing and creating digital content. (EV) Uses criteria to evaluate the quality of solutions, can identify improvements making some refinements to the solution, and future solutions. (EV)

Queries data on one table using a typical query language. (AB)

Knows that there is a range of operating systems and application software for the same hardware. (AB)

Evaluates the appropriateness of digital devices, internet services and application software to achieve given goals. (EV) Designs criteria to critically evaluate the quality of solutions, uses the criteria to identify improvements and can make appropriate refinements to the solution. (EV)

Digital Literacy

Understands the importance of communicating safely and respectfully online, and the need for keeping personal information private. (EV) Knows what to do when concerned about content or being contacted. (AL)

Knows common uses of information technology beyond the classroom. (GE) Shares their use of technology in school.

Demonstrates use of computers safely and responsibly, knowing a range of ways to report unacceptable content and contact when online.

Shows an awareness for the quality of digital content collected. (EV)

Recognises what is acceptable and unacceptable behaviour when using technologies and online services.

Makes judgements about digital content when evaluating and repurposing it for a given audience. (EV) (GE)

Demonstrates responsible use of technologies and online services, and knows a range of ways to report concerns.

Selects, combines and uses internet services. (EV)

Understands the potential of information technology for collaboration when computers are networked. (GE)

Recognises ethical issues surrounding the application of information technology beyond school.

<p>Defines data types: real numbers and Boolean. (AB) Knows that digital computers use binary to represent all data. (AB) Understands how bit patterns represent numbers and images. (AB) Knows that computers transfer data in binary. (AB) Understands the relationship between binary and file size (uncompressed). (AB)</p> <p>Recognises and understands the function of the main internal parts of basic computer architecture. (AB) Understands the concepts behind the fetch-execute cycle. (AB) (AL)</p> <p>Understands how search engines rank search results. (AL) Understands how to construct static web pages using HTML and CSS. (AL) (AB) Understands data transmission between digital computers over networks, including the internet i.e. IP addresses and packet switching. (AL) (AB)</p> <p>Understands a recursive solution to a problem repeatedly applies the same solution to smaller instances of the problem. (AL) (GE) Recognises that some problems share the same characteristics and use the same algorithm to solve both (generalisation). (AL) (GE) Understands the notion of performance for algorithms and appreciates that some algorithms have different performance characteristics for the same task. (AL) (EV)</p> <p>Uses nested selection statements. (AL) Appreciates the need for, and writes, custom functions including use of parameters. (AL) (AB) Knows the difference between, and uses appropriately, procedures and functions. (AL) (AB) Understands and uses negation with operators. (AL) Uses and manipulates one dimensional data structures. (AB) Detects and corrects syntactical errors. (AL)</p> <p>Understands how numbers, images, sounds and character sets use the same bit patterns. Performs simple operations using bit patterns e.g. binary addition. (AB) (GE) Understands the relationship between resolution and colour depth, including the effect on file size. (AB)</p> <p>Distinguishes between data used in a simple program (a variable) and the storage structure for that data. (AB)</p> <p>Understands the von Neumann architecture in relation to the fetch-execute cycle, including how data is stored in memory. (AB) (GE) Understands the basic function and operation of location addressable memory. (AB)</p>	<p>Knows the names of hardware e.g. hubs, routers, switches, and the names of protocols e.g. SMTP, IMAP, POP, FTP, TCP/IP, associated with networking computer systems. (AB)</p> <p>Justifies the choice of and independently combines and uses multiple digital devices, internet services and application software to achieve given goals. (EV)</p> <p>Evaluates the trustworthiness of digital content and considers the usability of visual design features when designing and creating digital artifacts for a known audience. (EV) Designs criteria for users to evaluate the quality of solutions, uses the feedback from the users to identify improvements and can make appropriate refinements to the solution. (EV)</p>	<p>Knows the purpose of the hardware and protocols associated with networking computer systems. (AB) (AL)</p> <p>Undertakes creative projects that collect, analyse, and evaluate data to meet the needs of a known user group. (AL) (DE) (EV) Effectively designs and creates digital artifacts for a wider or remote audience. (AL) (DE)</p> <p>Considers the properties of media when importing them into digital artifacts. (AB) Documents user feedback, the improvements identified and the refinements made to the solution. (AB)</p>	<p>Recognises that persistence of data on the internet requires careful protection of online identity and privacy.</p> <p>Explains and justifies how the use of technology impacts on society, from the perspective of social, economical, political, legal, ethical and moral issues. (EV)</p>
<p>Recognises and understands the function of the main internal parts of basic computer architecture. (AB) Understands the concepts behind the fetch-execute cycle. (AB) (AL)</p> <p>Understands how search engines rank search results. (AL) Understands how to construct static web pages using HTML and CSS. (AL) (AB) Understands data transmission between digital computers over networks, including the internet i.e. IP addresses and packet switching. (AL) (AB)</p> <p>Understands a recursive solution to a problem repeatedly applies the same solution to smaller instances of the problem. (AL) (GE) Recognises that some problems share the same characteristics and use the same algorithm to solve both (generalisation). (AL) (GE) Understands the notion of performance for algorithms and appreciates that some algorithms have different performance characteristics for the same task. (AL) (EV)</p> <p>Uses nested selection statements. (AL) Appreciates the need for, and writes, custom functions including use of parameters. (AL) (AB) Knows the difference between, and uses appropriately, procedures and functions. (AL) (AB) Understands and uses negation with operators. (AL) Uses and manipulates one dimensional data structures. (AB) Detects and corrects syntactical errors. (AL)</p> <p>Understands how numbers, images, sounds and character sets use the same bit patterns. Performs simple operations using bit patterns e.g. binary addition. (AB) (GE) Understands the relationship between resolution and colour depth, including the effect on file size. (AB)</p> <p>Distinguishes between data used in a simple program (a variable) and the storage structure for that data. (AB)</p> <p>Understands the von Neumann architecture in relation to the fetch-execute cycle, including how data is stored in memory. (AB) (GE) Understands the basic function and operation of location addressable memory. (AB)</p>	<p>Knows the names of hardware e.g. hubs, routers, switches, and the names of protocols e.g. SMTP, IMAP, POP, FTP, TCP/IP, associated with networking computer systems. (AB)</p> <p>Justifies the choice of and independently combines and uses multiple digital devices, internet services and application software to achieve given goals. (EV)</p> <p>Evaluates the trustworthiness of digital content and considers the usability of visual design features when designing and creating digital artifacts for a known audience. (EV) Designs criteria for users to evaluate the quality of solutions, uses the feedback from the users to identify improvements and can make appropriate refinements to the solution. (EV)</p>	<p>Knows the purpose of the hardware and protocols associated with networking computer systems. (AB) (AL)</p> <p>Undertakes creative projects that collect, analyse, and evaluate data to meet the needs of a known user group. (AL) (DE) (EV) Effectively designs and creates digital artifacts for a wider or remote audience. (AL) (DE)</p> <p>Considers the properties of media when importing them into digital artifacts. (AB) Documents user feedback, the improvements identified and the refinements made to the solution. (AB)</p>	<p>Recognises that persistence of data on the internet requires careful protection of online identity and privacy.</p> <p>Explains and justifies how the use of technology impacts on society, from the perspective of social, economical, political, legal, ethical and moral issues. (EV)</p>
<p>Recognises and understands the function of the main internal parts of basic computer architecture. (AB) Understands the concepts behind the fetch-execute cycle. (AB) (AL)</p> <p>Understands how search engines rank search results. (AL) Understands how to construct static web pages using HTML and CSS. (AL) (AB) Understands data transmission between digital computers over networks, including the internet i.e. IP addresses and packet switching. (AL) (AB)</p> <p>Understands a recursive solution to a problem repeatedly applies the same solution to smaller instances of the problem. (AL) (GE) Recognises that some problems share the same characteristics and use the same algorithm to solve both (generalisation). (AL) (GE) Understands the notion of performance for algorithms and appreciates that some algorithms have different performance characteristics for the same task. (AL) (EV)</p> <p>Uses nested selection statements. (AL) Appreciates the need for, and writes, custom functions including use of parameters. (AL) (AB) Knows the difference between, and uses appropriately, procedures and functions. (AL) (AB) Understands and uses negation with operators. (AL) Uses and manipulates one dimensional data structures. (AB) Detects and corrects syntactical errors. (AL)</p> <p>Understands how numbers, images, sounds and character sets use the same bit patterns. Performs simple operations using bit patterns e.g. binary addition. (AB) (GE) Understands the relationship between resolution and colour depth, including the effect on file size. (AB)</p> <p>Distinguishes between data used in a simple program (a variable) and the storage structure for that data. (AB)</p> <p>Understands the von Neumann architecture in relation to the fetch-execute cycle, including how data is stored in memory. (AB) (GE) Understands the basic function and operation of location addressable memory. (AB)</p>	<p>Knows the names of hardware e.g. hubs, routers, switches, and the names of protocols e.g. SMTP, IMAP, POP, FTP, TCP/IP, associated with networking computer systems. (AB)</p> <p>Justifies the choice of and independently combines and uses multiple digital devices, internet services and application software to achieve given goals. (EV)</p> <p>Evaluates the trustworthiness of digital content and considers the usability of visual design features when designing and creating digital artifacts for a known audience. (EV) Designs criteria for users to evaluate the quality of solutions, uses the feedback from the users to identify improvements and can make appropriate refinements to the solution. (EV)</p>	<p>Knows the purpose of the hardware and protocols associated with networking computer systems. (AB) (AL)</p> <p>Undertakes creative projects that collect, analyse, and evaluate data to meet the needs of a known user group. (AL) (DE) (EV) Effectively designs and creates digital artifacts for a wider or remote audience. (AL) (DE)</p> <p>Considers the properties of media when importing them into digital artifacts. (AB) Documents user feedback, the improvements identified and the refinements made to the solution. (AB)</p>	<p>Recognises that persistence of data on the internet requires careful protection of online identity and privacy.</p> <p>Explains and justifies how the use of technology impacts on society, from the perspective of social, economical, political, legal, ethical and moral issues. (EV)</p>
<p>Recognises and understands the function of the main internal parts of basic computer architecture. (AB) Understands the concepts behind the fetch-execute cycle. (AB) (AL)</p> <p>Understands how search engines rank search results. (AL) Understands how to construct static web pages using HTML and CSS. (AL) (AB) Understands data transmission between digital computers over networks, including the internet i.e. IP addresses and packet switching. (AL) (AB)</p> <p>Understands a recursive solution to a problem repeatedly applies the same solution to smaller instances of the problem. (AL) (GE) Recognises that some problems share the same characteristics and use the same algorithm to solve both (generalisation). (AL) (GE) Understands the notion of performance for algorithms and appreciates that some algorithms have different performance characteristics for the same task. (AL) (EV)</p> <p>Uses nested selection statements. (AL) Appreciates the need for, and writes, custom functions including use of parameters. (AL) (AB) Knows the difference between, and uses appropriately, procedures and functions. (AL) (AB) Understands and uses negation with operators. (AL) Uses and manipulates one dimensional data structures. (AB) Detects and corrects syntactical errors. (AL)</p> <p>Understands how numbers, images, sounds and character sets use the same bit patterns. Performs simple operations using bit patterns e.g. binary addition. (AB) (GE) Understands the relationship between resolution and colour depth, including the effect on file size. (AB)</p> <p>Distinguishes between data used in a simple program (a variable) and the storage structure for that data. (AB)</p> <p>Understands the von Neumann architecture in relation to the fetch-execute cycle, including how data is stored in memory. (AB) (GE) Understands the basic function and operation of location addressable memory. (AB)</p>	<p>Knows the names of hardware e.g. hubs, routers, switches, and the names of protocols e.g. SMTP, IMAP, POP, FTP, TCP/IP, associated with networking computer systems. (AB)</p> <p>Justifies the choice of and independently combines and uses multiple digital devices, internet services and application software to achieve given goals. (EV)</p> <p>Evaluates the trustworthiness of digital content and considers the usability of visual design features when designing and creating digital artifacts for a known audience. (EV) Designs criteria for users to evaluate the quality of solutions, uses the feedback from the users to identify improvements and can make appropriate refinements to the solution. (EV)</p>	<p>Knows the purpose of the hardware and protocols associated with networking computer systems. (AB) (AL)</p> <p>Undertakes creative projects that collect, analyse, and evaluate data to meet the needs of a known user group. (AL) (DE) (EV) Effectively designs and creates digital artifacts for a wider or remote audience. (AL) (DE)</p> <p>Considers the properties of media when importing them into digital artifacts. (AB) Documents user feedback, the improvements identified and the refinements made to the solution. (AB)</p>	<p>Recognises that persistence of data on the internet requires careful protection of online identity and privacy.</p> <p>Explains and justifies how the use of technology impacts on society, from the perspective of social, economical, political, legal, ethical and moral issues. (EV)</p>

Computational Thinking Concept: AB = Abstraction; DE = Decomposition; AL = Algorithmic Thinking; EV = Evaluation; GE = Generalisation

Note: Each of the Progression Pathway statements is underpinned by one-or-more learning outcomes (due for publication in 2014), providing greater detail of what should be taught to achieve each Progression Pathway statement and National Curriculum point of study.

© 2014 Mark Dorling and Matthew Walker. Reviewed by Simon Humphreys and Sue Sentence of Computing At School, CAS Master Teachers, and by teachers and academics from the wider CAS community.

Computing ≠ coding and programming

‘Coding is the new Latin’ has become a bit of a catchphrase and there is so much talk of coding and programming in the media that you might be forgiven for thinking that the main new element in the 2014 National Curriculum Computing Programmes of Study is programming.

Absolutely not! Programming plays the same role in computing that investigations do in maths or science. Programming animates the subject and brings computing to life; it is creative, and engaging. It illustrates otherwise abstract concepts in completely concrete terms. It is also an incredibly useful skill. Nevertheless computing is more than programming, just as chemistry is more than Bunsen burners and test tubes.

The first two aims of the 2014 National Curriculum Computing Programmes of Study say that all pupils:

can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation

can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems

Notice the emphasis on ‘fundamental principles’ and on ‘computational thinking’, both informed and illuminated repeatedly by programming. Essentially, programming bridges the gap between computational thinking and computers.

What exactly is ‘programming’ and is it the same as ‘coding’?

Here is an excerpt from a knitting pattern for a jumper:

A knitting pattern is very like a program:

- All the creativity is in the pattern. No creativity is required from the person or machine doing the knitting. They just knit, or execute, the pattern.
- The pattern is precise. The person or machine just has to follow the instructions and they will end up with a jumper.
- The pattern looks like gibberish to anyone who does not know how to knit. The pattern has its own ‘formal’ language, a language with a fixed grammar and vocabulary.
- Sophisticated patterns can be made by combining more than one instruction together and repeating combinations, even though the individual instructions are very simple.

Cast off 2 sts beg next 2 rows. 67 (**73-79-83-97-107-117-123**) sts.

1st row: (K1. P1) twice. S11. K1. psso. Knit to last 6 sts. K2tog. (P1. K1) twice.

2nd row: (P1. K1) twice. P2tog. Purl to last 6 sts. P2togtbl. (K1. P1) twice.

3rd row: As 1st row.

4th row: (P1. K1) twice. Purl to last 4 sts. (K1. P1) twice. Rep last 4 rows 0 (**0-1-1-0-3-6-6**) time(s) more. 61 (**67-67-71-91-83-75-81**) sts.








A ‘computer program’ is just a set of instructions that the computer executes in order to achieve a particular objective; and ‘programming’ is the craft of analysing problems, and designing, writing, testing and maintaining programs in order to solve them.

So what is the difference between ‘coding’ and ‘programming’? There is no settled consensus on what the difference is, or indeed whether there is one at all; different programmers might give you different answers. So it is safe, indeed advisable, to treat the two as synonymous, at least initially. It leaves much more time and brain space for more important questions. If and when you feel ready to engage with the discussion, there are posts on the Computing At School’s community pages (community.computingschool.org.uk/door) that may interest you.

Section 3: A road map for managing change

How do I create an inspiring and engaging curriculum?

Complete the *What is 'curriculum'? What is 'creativity'? What is innovation?* planning sheet individually, and then come together and agree departmental definitions. Next, watch the three videos and have a look at the associated resources, which illustrate what creative and innovative computing lessons look like. Then complete the *Our vision for a creative and innovative curriculum* planning sheet as a department.

		Book	CD-ROM	Website
	Planning sheet: <i>What is 'curriculum'? What is 'creativity'? What is innovation?</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.	✓ Page 22	✓	✓
	Video: <i>Teaching computing through dance</i> © Crown Copyright 2013. This content is available under the Open Government Licence v2.0.		✓	✓
	Activity: <i>Creating an animation (Teaching computing through dance)</i> © Sarah Lawrey 2014.		✓	✓
	Video: <i>Beating the clock – sorting networks</i> © CSUnplugged. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.		✓	✓
	Activity: <i>Beating the clock – sorting networks</i> © CSUnplugged. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.		✓	✓
	Video: <i>Teaching computing to all</i> © Hodder Education 2014. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.		✓	✓
	Planning sheet: <i>Our vision for a creative and innovative curriculum</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.	✓ Page 23	✓	✓

What do my students need and how much time do I have to teach them?

Using the *What prior knowledge do students starting GCSE need?* information sheet to help you consider the implications for students starting the KS3 National Curriculum Programme of Study for Computing in Year 8 or Year 9, identify what each year group needs and the time you have available to teach them. Remember that what you do at Key Stage 4 – delivering the statutory requirements to those students not following a Level 2 qualification and teaching Level 2 qualifications in Computer Science or Information Technology to those who are – will impact on your staffing and room allocation at Key Stage 3, as will your vision for an innovative and creative curriculum.

		Book	CD-ROM	Website
	Information sheet: <i>What prior knowledge do students starting GCSE need?</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.	✓ Page 24	✓	✓

Do I need to change my entire Key Stage 3 schemes of work or can I reuse aspects of my Key Stage 3 ICT schemes of work?




Review your existing schemes of work against your vision for an innovative and creative curriculum:

- Ask yourself, 'Is what we're offering here what we want to offer in the future?' Flag those aspects of your schemes of work that **do** fit with your vision or are useful even if they don't set the world alight; you can consider later whether they demonstrate computational thinking and meet the requirements of the 2014 Key Stage 3 National Curriculum Programmes of Study for Computing.
- Get rid of those aspects of your schemes of work and associated activities that **do not** fit with your vision. Hopefully you'll be motivated to seek out activities to take their place, once you see that you are not throwing out all your schemes of work and are able to develop those activities you know your students enjoy and get value from.

How do I pull together new schemes of work?



Using the *Planning Key Stage 3 spreadsheet*, identify which aspects of the Key Stage 3 curriculum are covered by the bits of your existing schemes of work you are keeping. Then use the *Interactive Progression Pathways Tool* and Section 5 of this toolkit to fill in the gaps in your schemes of work and to ensure that all your activities encourage computational thinking. Remember to consider progression as you compile your schemes of work. The *How to develop schemes of work* information sheet provides further guidance on this process.

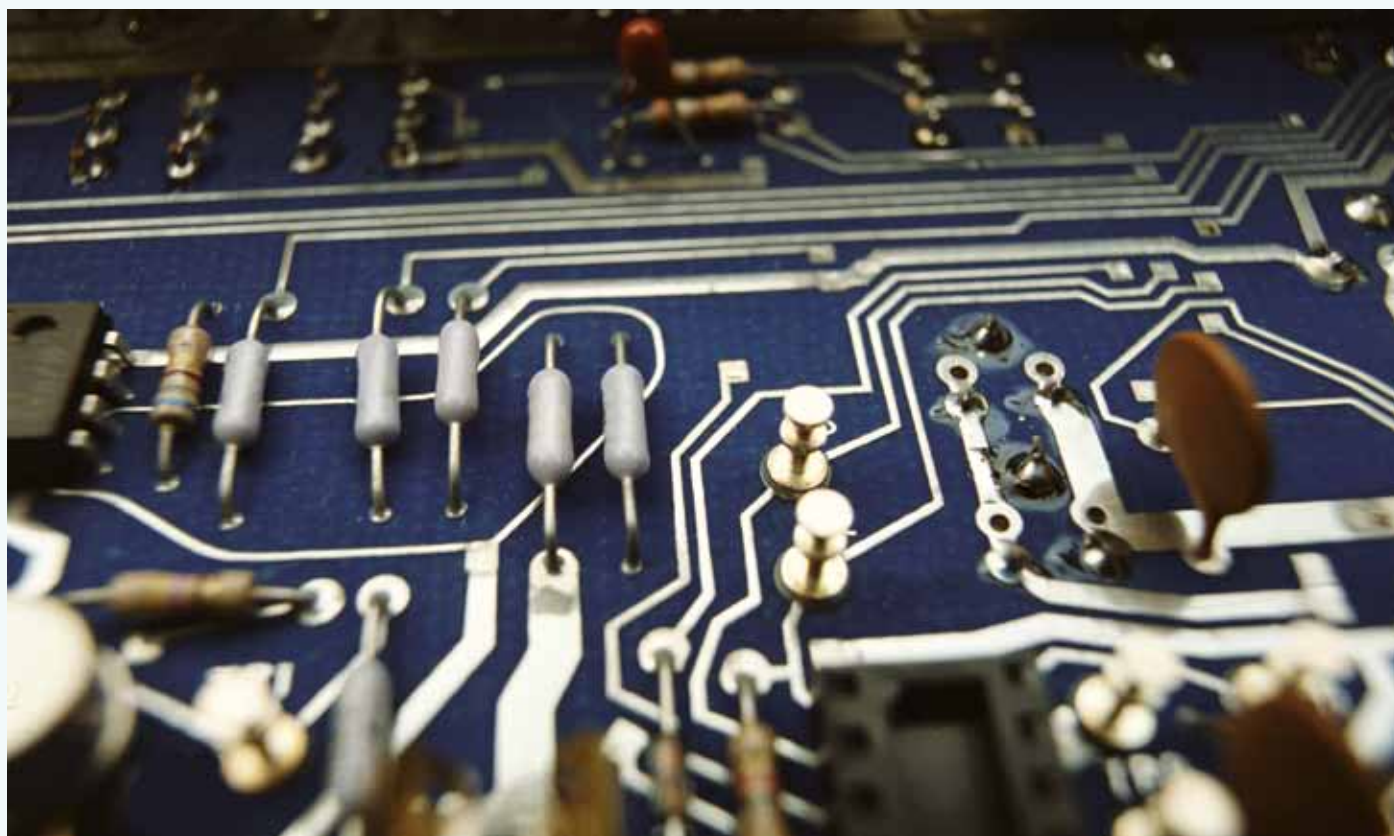
The *Interactive Progression Pathways tool* is an interactive website that links the National Curriculum Programmes of Study bullet points, the Progression Pathways statements and the Computational Thinking Framework statements, and generates a list of associated teaching and learning resources. Once a list of resources has been generated you can share it using a unique weblink or email it to yourself or others.

		Book	CD-ROM	Website
	Planning sheet: <i>Planning Key Stage 3 spreadsheet</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.		✓	✓
	Interactive: <i>Interactive Progression Pathways tool</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.		✓	✓
	Information sheet: <i>How to develop schemes of work</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.	✓ Page 25	✓	✓

Use the *Time to reflect 2* planning sheet to consider the effects that implementing the new curriculum will, or could, have on you and your school and then, as a department, agree the milestones for the next 12 months using the *Milestones* planning sheet.

Finally, update and add to the presentation you created at the end of Section 2 so that it clearly presents your vision for a creative and innovative curriculum underpinned by computational thinking, and explains how you are going to set about fulfilling this vision. Your presentation may need to acknowledge that some staff will find the changes daunting and will need extra support. Remember that this is a two or three year process, and your vision may develop and change over time, so consider how you will share information with the Senior Leadership Team and colleagues in other departments in the future. You may also want to take this opportunity, if you haven't already done so, to write to parents and carers to explain what is happening.

		Book	CD-ROM	Website
	Planning sheet: <i>Time to reflect 2</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.	✓ Page 26	✓	✓
	Planning sheet: <i>Milestones</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.	✓ Page 27	✓	✓



What is 'curriculum'? What is 'creativity'? What is 'innovation'?

Imagine you are walking down a corridor in your school in 12 or 24 months' time and you look into a classroom. An innovative and creative computing lesson is taking place. What does it look like? What is the teacher doing? What are the students doing?

First, use the space below to write a definition of '**curriculum**':

For example, a curriculum is a structured programme of learning designed to achieve designated educational outcomes.

Next, use the space below to define '**creativity**':

For example, creativity is the process of creating something unique and valuable.

Then, define '**innovation**':

For example, innovation is finding a better way to do something.

Our vision for a creative and innovative curriculum

A vision statement provides you with a map that helps you reach your destination. Regularly analysing what you are doing to ensure it contributes towards realising your vision can be a strong motivational tool and will ensure that your efforts are focused in the right direction.

Here is an example of a vision statement for a creative and innovative computing curriculum. Underline the sections that you think are most important.

The curriculum will be broad, balanced and integrated, with the breadth and depth that will support students whichever pathway they choose at Key Stage 4 and beyond. We will provide students with an understanding of the foundations of computing, to help them better apply information technology and better understand the implications of the technologies they use. Wherever possible we will teach computing without computers, but will use information technology to evidence and enhance learning. We will use computational thinking to develop confident, creative and resilient problem solvers. Teaching and learning will focus on solving (scalable) real-life problems, and we will ensure strong links to other curriculum subjects. We will use programming tools to bridge the gap between models (good computational thinking) and computers, selecting the appropriate programming language for the challenge and enabling us to teach key programming concepts rather than teaching programming languages for the sake of it.

Use the space below to write a vision statement for the computing curriculum you will deliver in your school.

What prior knowledge do students starting GCSE need?

If students are to be successful at GCSE, acquiring basic programming and computational thinking skills at Key Stage 3 is vital.

Programming is a common element in all the GCSE specifications. Candidates must be able to analyse, design, code, test and evaluate a program written in a high level programming language, such as Pascal, Python or Small Basic. In the National Curriculum Computing Programme of Study for Key Stage 3 this is:

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 (for example, lists tables or arrays); design and develop modular programs that use procedures or functions.

Designing a solution to a programming problem requires an algorithmic approach and an appreciation of key algorithms and computational thinking is therefore essential. Being able to write and follow algorithms is also part of the examined content for all three major specifications at GCSE. In the National Curriculum Computing Programme of Study for Key Stage 3 this is:

understand several key algorithms that reflect computational thinking (for example, one for sorting and searching); to use logical reasoning to compare the utility of alternative algorithms for the same problem.

All three major GCSE specifications require an understanding of the basics of Boolean algebra and the use of AND, OR and NOT in logic circuits and in programming. In the National Curriculum Computing Programme of Study for Key Stage 3 this is:

understand simple Boolean logic (for example, AND, OR and NOT) and some of its uses in circuits and programming.

An understanding of the basic hardware and software that makes up a computer system is

fundamental to all GCSE specifications. In the National Curriculum Computing Programme of Study for Key Stage 3 this is:

understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems.

An appreciation of the various data types and how computers store and use data in binary is central to all three major specifications at GCSE. They all require the candidate to be able to work with binary to perform simple calculations and conversions. In the National Curriculum Computing Programme of Study for Key Stage 3 this is:

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

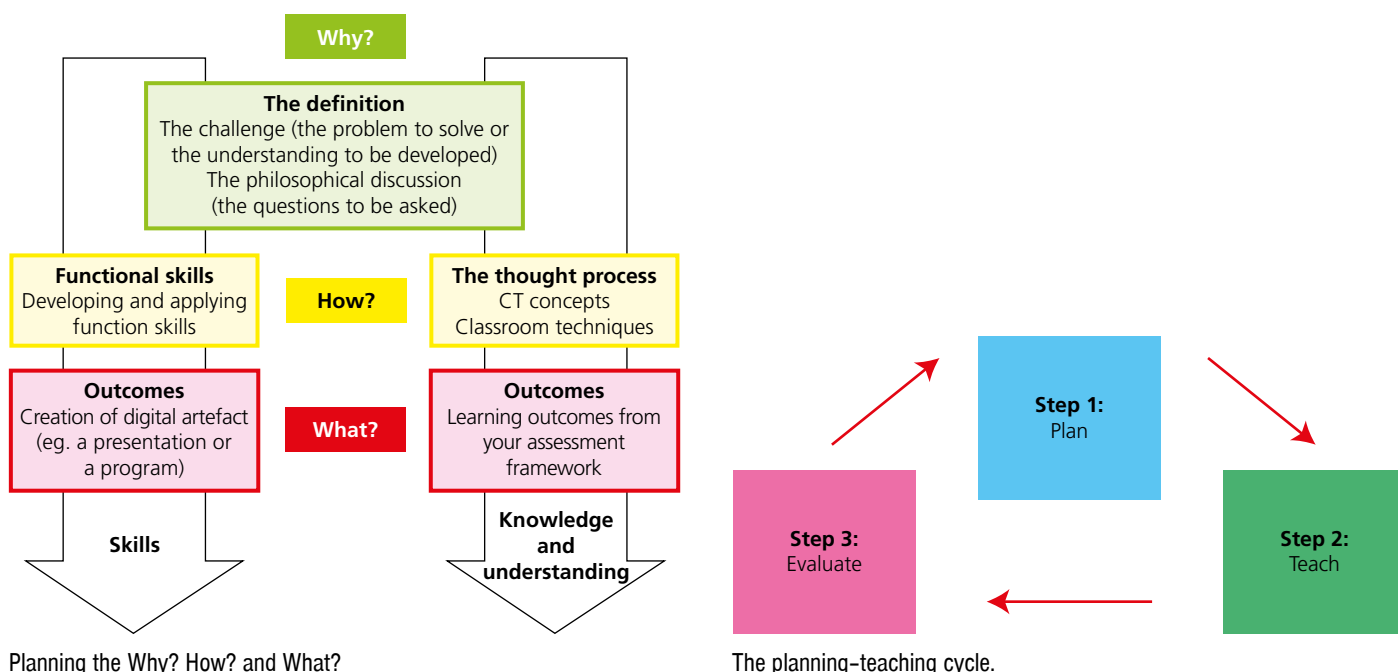
understand how numbers can be represented in binary, and be able to carry out simple operations on binary numbers (for example, binary addition, and conversion between binary and decimal).

In order to prepare students for GCSE they need to be introduced to and have grounding in all of these areas but learning to program takes time and should be a key element of any Key Stage 3 course. Remember, though, that computing is more than just programming and that computational thinking skills are at the heart of computing. Learning to program starts with the core computational thinking skills of decomposition, abstraction, algorithm design, generalisation and evaluation. After all, you need to understand the problem and identify a solution before you can begin programming. So, if there are any skills that a student should have at their disposal when starting out on a GCSE course they are these.

* Remember, if you need further guidance on the breadth and depth expected for each of the bullet points in the National Curriculum Programmes of Study, look at *Computing in the National Curriculum: a guide for secondary teachers*. The *Interactive Progression Pathways Tool* will also help you.

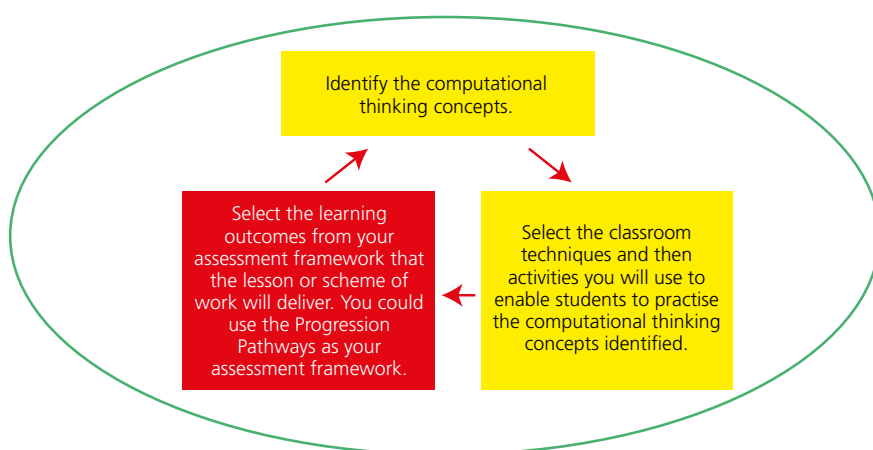
How to develop schemes of work

We all develop schemes of work differently and there is no right or wrong way to go about the task. However, a well-thought-out scheme of work will make it clear to students **why** they are doing what they are doing, **how** they should be doing it and **what** they will be learning and/or producing. This will be done as part of Step 1 of the planning–teaching cycle.



We start with the 'Why?' What is the hook that will excite the students, the problem you will set them to solve, the big question you will ask them to answer or the understanding you want them to develop? Listen to students, discuss cross-curricular opportunities with colleagues and look at *cs4fn* (www.cs4fn.org) for inspiration. STEM ambassadors, code clubs, academics and people in business are also a really useful source of support as they are often able to suggest real-life problems that will inject something inspiring into your schemes of work, and they may be willing to help you teach them too!

Then we tackle the 'How?' and the 'What?', moving through the following steps until we have created a series of lesson plans:



You can use the *Interactive Progression Pathways Tool* to guide this process.

Time to reflect 2

Implementing the new curriculum presents exciting opportunities and many challenges. Take some time to reflect on the effects that it will or could have by completing the table below. You could consider staffing and CPD, room allocation, cross-curricular teaching and learning, extra curricular clubs and STEM ambassadors.

What effects do I already know I will experience? What facts do I have to evidence this?	
What effects do I think I will experience?	
What benefits will the changes have?	
Are there any potential problems that will result from the changes?	
Can you increase the benefits and decrease the potential problems by doing anything differently?	

Milestones

It takes time to implement a new curriculum and it is important to have a plan to guide your work in the short, medium and long term. Use the tables below to agree your milestones for the next 12 months. Your milestones should include some or all of the tasks we have suggested you complete.

In three months we will have:

A brief description of what will have been done	The name of the person or people who will do it	What will the outcome look like?

In six months we will have:

In nine months we will have:

In 12 months we will have:

Section 4: Teaching

Section 4 should be used in conjunction with Section 5: Resources and, particularly, the *Interactive Progression Pathways Tool*, which contains links to a wealth of great resources.

Which programming languages should I use?

Although, as you know, programming isn't the be all and end all of computing, it is an important aspect of the 2014 curriculum and you will need to decide which programming languages you are going to use. The *Choosing a programming language* video offers six key questions you should ask yourself before making your decision.

		Book	CD-ROM	Website
	Video: <i>Choosing a programming language</i> © Hodder Education 2014. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.		✓	✓

What does a good computing lesson look like?

The *Lesson observation form with prompts* planning sheet provides a pro-forma, based on Ofsted's Evidence Form for school inspections, that can be used during a lesson observation to ensure a focus on computational thinking, and the *Completed lesson observation form* information sheet is a completed example. A link to Ofsted's *Note to inspectors: use of assessment information during inspections in 2014/15* is also provided.








And, of course, don't forget everything you already know about teaching a great lesson, including ensuring there is progression and providing students with an opportunity to recap their learning.

There is a range of techniques for teaching computing and several are presented here:

- The *Orange Game* video from CSUnplugged illustrates how theory can be taught away from a computer using kinesthetic activities.
- The *Paired programming* video shows how students can collaborate and develop resilience whilst writing and debugging their programs.
- The *From graphical to text-based programming languages* information sheet illustrates how to effectively manage the transition from graphical to text-based programming languages.

Techniques are constantly evolving. Please take an active part in the discussions at CAS online to keep up with the latest developments.



The *Creative computing in action* video illustrates how all this can be put into practice in the classroom.

		Book	CD-ROM	Website
	Planning sheet: <i>Lesson observation form with prompts</i> © Crown Copyright 2014. Contains public sector information licensed under the Open Government Licence v3.0.	✓ Page 30	✓	✓
	Information sheet: <i>Completed lesson observation form</i> © Crown Copyright 2014. Contains public sector information licensed under the Open Government Licence v3.0.		✓	✓
	Link: <i>Note to inspectors: use of assessment information during inspections in 2014/15</i>		✓	✓
	Video: <i>The Orange Game</i> © CSUnplugged. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.		✓	✓
	Video: <i>Paired programming</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.		✓	✓
	Information sheet: <i>From graphical to text-based programming languages</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.	✓ Pages 31-2	✓	✓
	Video: <i>Creative computing in action</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.		✓	✓

How do I structure a good computing lesson?

The *Lesson plan template* highlights the key things to think about when planning a lesson and provides a structure that you can adapt to meet your needs and the needs of your students. We have also included a scheme of work from the Digital Schoolhouse, showing the lesson plan template in action. Of course, as your subject knowledge and expertise grows over time, you can simplify the planning process but it is important to ensure that you continue to emphasise the links to computational thinking as well as progression.

While planning your lessons, think about the impact of the activities on your classroom layout. For example, do you need to push the desks to the wall to create enough space to allow your students to get the most out of an unplugged activity?

		Book	CD-ROM	Website
	Information sheet: <i>Lesson plan template</i> © Digital Schoolhouse 2014. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.	✓ Pages 33-4	✓ With and without speech bubbles	✓ With and without speech bubbles
	Activity: <i>Shapes calculator scheme of work</i> © Digital Schoolhouse 2014. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.	✓	✓	✓

Lesson observation form with prompts

Evidence form – S5											
Inspection No		Inspector's OIN		Date		Time of day		EF No			
				/ /							
Observation type (please tick one box only)											
Lesson observation				Work analysis				Discussions			
Other											
Focus (inspection trail or main purpose of the activity)				Context (lesson objective or description of activity)							
Information gathered for lesson observations only											
Year group (s)				Grouping (see footnote ¹)		MC SU SA SL O		Gender		Subject codes	
								B G MI		IT	
										Present /NOR	
<p>Evidence</p> <p>Was the topic in this lesson published on the curriculum content section of the school's public website?</p> <p>Does this topic represent a progression/challenge in the children's understanding of the principles of computing through evidence from a review of children's work from previous lessons?</p> <p>Do the lesson objectives introduce or revisit aspects of computational thinking (e.g. autonomous execution of pre-designed instructions)?</p> <p>Do the lesson activities practise different skills of computational thinking (e.g. algorithmic thinking)?</p> <p>Are there enough challenges for gifted and talented children?</p> <p>Are the children aware of, and can they elaborate on, the knowledge and understanding gained in this lesson?</p> <p>Are the children encouraged to introduce their own questions to the lesson?</p> <p>Are the children responding quickly to the teacher's instruction and requests, allowing the lesson to flow smoothly and without interruption?</p> <p>Evidence of SMSC</p> <p>How do the children handle disagreements in discussion/group work?</p> <p>Are the children applying a wide range of knowledge and understanding from their day-to-day experience, including that from their cyber world?</p> <p>Are the children reflective in their learning and developing their curiosity over time?</p> <p>Evaluation</p> <p>Are the children properly prepared for the lesson such that they are ready and eager to learn?</p> <p>Do the teacher and other adults have high expectations and high aspirations to expand their own and children's knowledge of computing?</p> <p>Does the teaching engage and include all children with work that is challenging enough for different groups?</p> <p>Does the teacher reshape tasks and explanations during the lesson in response to children's understanding?</p>											
Use for grades if there is sufficient evidence:								Time spent in this lesson (minutes)			
Achievement of pupils				Quality of teaching				Running EF?		Y	N
Behaviour and safety of pupils				Leadership and management				Number of lessons included in running EF			
NQT				ITE route				If yes, cumulative time (minutes)			
ITE provider								Special focus, complete if necessary			

September 2014

¹ Grouping codes: MC = Mixed ability class; SU = Setted, upper ability; SA = Setted, average ability; SL = Setted, lower ability; O = Other

From graphical to text-based programming languages

At Key Stage 3 there is a requirement that:

Pupils should be taught to:

- use two or more programming languages, at least one of which is textual, to solve a variety of computational problems

There is a perception amongst non-specialist teachers that text-based programming is hard. This view is not without foundation if students have to wrestle with the technical details of a new programming language at the same time as they are identifying a problem and developing a program to solve it.

However, utilising pedagogical approaches from other subjects can make the transition from a graphical programming language to a text-based programming language easier to teach and, consequently, easier to learn. Experience has also shown that, once students understand the basics of text-based programming languages, they actively want to use them because they see how limiting graphical programming languages are.

Learning from cross-curricular pedagogical approaches

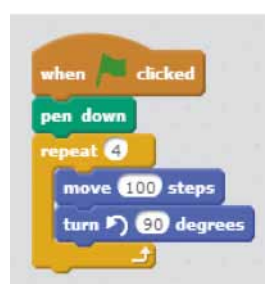
Teachers, especially at primary level and in modern foreign language teaching, introduce a range of strategies to support the development of literacy skills by students who have varying backgrounds, experiences and levels of linguistic ability. Teachers will use students' existing knowledge of language systems to develop further awareness of how language works. They will also encourage an understanding of the spelling patterns of the language and sound–symbol relationships to support word level awareness. These approaches reinforce and utilise students' existing uses of language and literacy skills. Teachers will facilitate linguistic development in sentence level writing skills where learners demonstrate their word level understanding in meaningful contexts and purposes. Teachers are also aware of the importance of maintaining a balance between improving students' writing sub-skills and enhancing compositional skills (text-level skills) using cohesive and text organisation devices effectively.

We can transfer some of these approaches into the teaching of text-based programming languages. For example, students begin by reading graphical programming constructs and blocks of code, commenting the code to explain what is happening, and then match the graphical programming constructs and blocks of code to their corresponding text-based programming constructs and excerpts of code. This helps them develop an ability to read and comprehend the text-based programming language. Taking a very simple example to illustrate the point:



```
from turtle import *
#
fd(100)
lt(90)
fd(100)
lt(90)
fd(100)
lt(90)
fd(100)
lt(90)
```

Drawing a square in Scratch and drawing a square in Python.



```
from turtle import *
#
for turn in range(4):
    fd(100)
    lt(90)
```

Drawing a square using the repeat block in Scratch and the 'for' loop in Python.

The graphical blocks of code are slowly removed and students are asked to draw or program those that are missing. When students have developed their confidence in reading the text-based code through this activity, the focus is changed. The graphical blocks of code are reintroduced and the excerpts of text-based code are slowly removed, and students are asked to write or program those that are missing. Leaving some excerpts of text-based code visible provides support for less able students.

Applying the pedagogy

This same pedagogy can be used when students take on new and unfamiliar challenges. For example, students could be presented with the challenge of using a text-based programming language to program a robot that navigates around a maze, and their learning could be scaffolded as follows:

1. Students create a paper net and a paper maze and develop their algorithm unplugged.
2. Students develop and debug a program in Scratch.
3. Students develop and debug a program in a text-based programming language using, for example, RoboMind.

The solution to the problem is reached through several iterations, with students developing their confidence, independence and resilience at each stage.

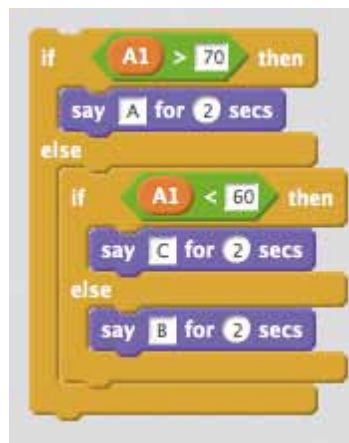
It can even be used to support students' digital literacy skills. Students often struggle to combine two 'if else' statements into a single nested 'if else' statement in a spreadsheet. To help them understand what they need to do, you could model the process in a graphical programming language and then take students' new-found understanding and apply it to a spreadsheet, drawing out the similarities between, for example, Scratch and a spreadsheet formula.



=IF(A1>70, "A", "B")

=IF(A1<60, "C", "B")

Two 'if else' statements in Scratch and Excel.



=IF(A1>70, "A",

IF(A1<60, "C", "B"))

or

=IF(A1>70, "A",
IF(A1<60, "C", "B"))

A single nested 'if else' statement in Scratch and Excel.

Action research

These findings emerged from an action research project and you can find out more about the project by having a look at the following conference paper: Dorling, M. and White, D. 'Scratch: a way to Logo and Python', to be presented at the 46th SIGCSE Technical Symposium in 2015. As the subject of computing grows in confidence in schools, more ideas like this will emerge. Try out some action research yourself, perhaps in partnership with local academics, and see what you can add to the conversation. What can you pinch from other subjects to help you make computing more accessible for your students?

Lesson plan template

Topic	Class

Prior learning/place of lesson in scheme of work

It's a good idea to consider what students have already done that will help them in this lesson. Does this lesson tie in with any previous work? Is it part of a sequence of lessons? Even if this is the first lesson in a sequence, do you expect any prior knowledge, perhaps from previous years?

What the teacher needs to know

This is especially important if you are developing the lesson plan for your department. Teachers who are still relatively new to computing need to know the level of skills and understanding they have to have in order to teach the lesson effectively, and this information is not always clear when you look at the activities.

Resources

Objectives

If your lesson includes cross-curricular learning, reflect this in the objectives. Talk to your colleagues too, because other subjects can provide useful content and realistic problems for your computing students to solve.

Links to the National Curriculum Programmes of Study for Computing

Consider all the Key Stages. For example, just because you are teaching Key Stage 3 doesn't mean you aren't able to cover part of the Key Stage 4 curriculum with very little adaptation, if you study the bullet points closely.

Assessment

Progression Pathways

Computational Thinking Framework

Reference the elements of the Progression Pathways and the Computational Thinking Framework that are covered by this lesson. By identifying these from the start you will be able to plan an assessment strategy for a scheme of work, as well as easily identify opportunities to extend your more able students.

Time	Activity	Supporting different student groups	Assessment opportunities

Creating a new row for each activity makes the lesson plan much easier to read and makes it easier to plan timings, differentiation and assessment. Consider including alternative teaching styles, mixing up teacher-demonstrated activities with student investigations.

Homework	Intended follow on

What will students learn in future lessons? How could this lesson be modified and extended? If this lesson is part of a sequence of lessons, what happens next?

Section 5: Resources

What makes an effective activity?

First and foremost an effective computing activity develops computational thinking. *What makes an effective activity?* is a hands-on activity designed to help you spot the computational thinking in activities and add computational thinking to well-loved activities that you currently use but which are missing the all-important ingredient.

You can also use the *Interactive Progression Pathways tool* to help you identify computational thinking. The tool is an interactive website that links the National Curriculum Programmes of Study bullet points, the Progression Pathways statements and the Computational Thinking Framework statements, and generates a list of associated teaching and learning resources. You can match an activity you already have to a Progression Pathways statement and then use the tool to help you identify how many statements from the Computational Thinking Framework apply to that Progression Pathways statement. You can then see how many of the Computational Thinking Framework statements your activity ticks off.

		Book	CD-ROM	Website
	Activity: <i>What makes an effective activity?</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.	✓ Pages 36-44	✓	✓
	Interactive: <i>Interactive Progression Pathways tool</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.		✓	✓

Where do I find high quality resources?

The *Interactive Progression Pathways Tool* contains a whole of host of resources mapped to the Progression Pathways statements. The *Resources and courses* information sheet also contains a list of non-commercial and commercial organisations that provide high quality resources and courses to support the teaching and learning of computing.

		Book	CD-ROM	Website
	Interactive: <i>Interactive Progression Pathways tool</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.		✓	✓
	Information sheet: <i>Resources and courses</i> Text, logos and images © relevant organisations.	✓ Pages 45-52	✓	✓

What makes an effective activity?

Part one

Look at the three computing activities – Activity 1: General election on pages 40–1, Activity 2: Game of NIM on pages 42–3 and Activity 3: Using Scratch to create an interactive game on page 44 – and consider how you would teach each of them in your classroom and the outcomes you would expect.

The introductory statement of the 2014 National Curriculum Programmes of Study for Computing is, 'A high-quality computing education equips students to use computational thinking and creativity to understand and change the world.' Consider how the activities ensure that students develop their computational thinking.

Computational thinking is the thought processes or thinking skills associated with abstraction, algorithmic thinking, decomposition, evaluation and generalisation. These thinking skills are associated with the activities of people used to working in a computing environment. However, they are also generic and help all students see the world and analyse complex artefacts, processes or systems in a rigorous and precise manner.

Abstraction is reducing complexity by removing unnecessary detail.

Decomposition is breaking a complex artefact, process or system into its component parts.

Algorithmic thinking is identifying and/or creating the sequence or rules associated with a process or system.

Evaluation is precise and rigorous assessment based on specific criteria, heuristics (rules that are only loosely defined) and users' needs.

Generalisation is identifying patterns and similarities in applications, solutions and data structures and using this knowledge in new contexts.

To help you complete this activity, consider which statements from the Progression Pathways and the Computational Thinking Framework can be added to the table opposite. You could use the Interactive Progression Pathways Tool to help you.

Progression Pathways		Computational Thinking Framework						
	All students	Most students	Some students	AB	DE	AL	EV	GE
Activity 1: General election								
Activity 2: Game of NIM								
Activity 3: Using Scratch to create an interactive game								

AB = Abstraction; DE = Decomposition; AL = Algorithmic thinking; EV = Evaluation; GE = Generalisation

Part two

In the space below, describe how Activity 1 and Activity 2 develop students' computational thinking skills.

Activity 1: General election	
Abstraction	
Algorithmic thinking	
Generalisation	

Activity 2: Game of NIM	
Abstraction	
Decomposition	
Algorithmic thinking	

Part three

Activity 1 is based on a lesson that existed in the curriculum in 2001 and it shows that previous schemes of work and activities provide opportunities for computational thinking.

The plenary would include a discussion of the realism of the models, with the teacher explaining that:

- The model is an abstraction of real life. Only the important details are entered. The rules for the model (the formulae) are an algorithm.
- By adapting the structure of one model and applying it to another scenario we are generalising.
- The greater the number of variables and the more accurate the formula, the more likely it is that the model will produce accurate predictions.
- Models like this are used to determine government policy, financial investments, planning for events, sales strategies for big retailers such as supermarkets and to predict the weather.

Activity 1 is effective because it enables students to 'design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems'

and to ‘create, reuse, revise and repurpose digital artefacts for a given audience, with attention to trustworthiness, design and usability’.

Activity 1 is effective because it promotes computational thinking – ‘A high-quality computing education equips students to use computational thinking and creativity to understand and change the world’ – and it also provides an opportunity for students to ‘evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems’.

Activity 2 is a computing lesson that does not require a computer. It is unplugged and demonstrates that students do not have to sit in front of a computer writing programs to be learning the broader computer science curriculum.

The plenary would involve a discussion of the game and the strategy used to win. The teacher would explain:

- How students had to think about the numbers, identifying what each pile of counters represented as an abstraction (9 counters were $8 + 1$, 7 counters were $4 + 2 + 1$ etc.).
- That an algorithm is not only a sequence of instructions but also the rules by which computers (and humans) operate and that the strategy used to win NIM is an algorithm.
- Playing the game is a complex activity but using decomposition – breaking the whole into component parts (turns) – the problem can be more easily solved.

This activity is effective because it enables students to ‘design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems’; because it promotes computational thinking (‘A high-quality computing education equips students to use computational thinking and creativity to understand and change the world’); and because it provides an opportunity for students to ‘understand how numbers can be represented in binary, and be able to carry out simple operations on binary numbers’.

As it stands, the third activity does not lead to higher level computational thinking. It is a copying exercise. Computer programming is not the objective of the 2014 National Curriculum Programmes of Study for Computing. It is one way, a very good way, of enabling students to develop their computational thinking skills but there are many other ways, including unplugged activities and those associated with spreadsheets.

Consider how to make Activity 3 a valid computational thinking activity and describe how your recast activity includes abstraction, evaluation and generalisation:

Activity 3: Using Scratch to create an interactive game	
Abstraction	
Evaluation	
Generalisation	

Activity 1: General election

Introduce the concept of modelling, creating something that imitates a real-life situation, and the idea that a computer-based model can be an aircraft simulator, a driving game or an avatar-explored virtual environment. Typing 'computer simulation' into Wikipedia will bring up many more examples of computer-based models.

The spreadsheet General_Election.xlsx is an interactive model that has been constructed to represent the way in which people might vote in a general election and thereby to predict the outcome of the election. The figure on the bottom right is the number of people expected to vote for the opposition and if it reaches 50% then the current government will fall. Open the spreadsheet.

	A	B	C	D	E	F	G	H
1			baseline	current	factored	weighting	weighted	
2	NHS	50 poll satisfaction	50	55	-500	1	-500	
3	Unemployment	BBC 2 million	2000000	2000400	400	1	400	
4	Wages	Daily Mirror £26.50	26500	26500	0	1	0	
5	Inflation RPI	BBC 2.5%	2.5	2.5	0	1	0	
6							-100	
7		About 50% vote for opposition	50				44.9	Government retains power

Talk about what influences the way people vote and, importantly, what might cause them to change the way they vote. In this model, these factors are limited to four:

1. The confidence people have in the way the NHS is being supported by the government
2. The rate of unemployment
3. The wages people receive
4. The rate of inflation.

Talk about how changes in unemployment would influence voting patterns. Then change the 'current' unemployment value. How does the outcome of the election change?

	A	B	C	D	E	F	G	H
1			baseline	current	factored	weighting	weighted	
2	NHS	50 poll satisfaction	50	55	-500	1	-500	
3	Unemployment	BBC 2 million	2000000	2000400	400	1	400	
4	Wages	Daily Mirror £26.50	26500	26500	0	1	0	
5	Inflation RPI	BBC 2.5%	2.5	2.5	0	1	0	
6							-100	
7		About 50% vote for opposition	50				44.9	Government retains power

When students understand what the model does, show them the formulae that drive the model.

General election ☆							
File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive							
fx							
	A	B	C	D	E	F	G
1			baseline	current	factored	weighting	weighted
2	NHS	50 poll satisfaction	50	55	$= (50 - D2) * 100$	1	$= E2 * F2$
3	$= \text{HYPERLINK}("http://www.bbc.com/news/health-2000000")$	$= \text{HYPERLINK}("http://www.bbc.com/news/health-2000000")$	2000000	2000400	$= (D3 - 2000000) * 1$	1	$= E3 * F3$
4	Wages	$= \text{HYPERLINK}("http://www.bbc.com/news/health-26500")$	26500	26500	$= (C4 - D4) * 2$	1	$= E4 * F4$
5	$= \text{HYPERLINK}("http://www.bbc.com/news/health-2.5")$	$= \text{HYPERLINK}("http://www.bbc.com/news/health-2.5")$	2.5	2.5	$= (D5 - C5) * 100$	1	$= E5 * F5$
6							$= \text{SUM}(G2:G5)$
7		About 50% vote for opposition	50			$= 45 + (G6 / 1000)$	$= \text{IF}(G7 > 45, "Opposition wins the election", "Government retains power")$

Explain how the NHS formula works so that the lower the figure in D2 becomes, the higher the value in G7 becomes and the more likely it is that the government will fall.

General election ☆							
File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive							
fx							
	A	B	C	D	E	F	G
1			baseline	current	factored	weighting	weighted
2	NHS	50 poll satisfaction	50	55	$= (50 - D2) * 100$	1	$= E2 * F2$
3	$= \text{HYPERLINK}("http://www.bbc.com/news/health-2000000")$	$= \text{HYPERLINK}("http://www.bbc.com/news/health-2000000")$	2000000	2000400	$= (D3 - 2000000) * 1$	1	$= E3 * F3$
4	Wages	$= \text{HYPERLINK}("http://www.bbc.com/news/health-26500")$	26500	26500	$= (C4 - D4) * 2$	1	$= E4 * F4$
5	$= \text{HYPERLINK}("http://www.bbc.com/news/health-2.5")$	$= \text{HYPERLINK}("http://www.bbc.com/news/health-2.5")$	2.5	2.5	$= (D5 - C5) * 100$	1	$= E5 * F5$
6							$= \text{SUM}(G2:G5)$
7		About 50% vote for opposition	50			$= 45 + (G6 / 1000)$	$= \text{IF}(G7 > 45, "Opposition wins the election", "Government retains power")$

Let students explore the model and ask 'What if?' questions.

Introduce the idea of a local football team (or a similar context that would be of interest to your students). What factors might influence their success at the end of the season? Perhaps they will suggest:

- The length of the summer break training abroad
- The number of supporters, in terms of money taken at the turnstile
- The amount of money spent on new players.

Ask students to use General_Election.xlsx to create a model for the local football team. They can change the labels and adapt the formulae accordingly.

Activity 2: Game of NIM

Remind students that the binary representation of numbers uses base 2. For example, you count 1, 2, 4, 8, 16 etc. in binary and

$$29 = 0 \times 32 + 1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$$

$$= 011101$$

Introduce the game of NIM:

- It is a game of strategy
- It is a two player game
- You divide the counters (say 30) into between four and seven piles of random sizes
- Each player takes it in turns to remove one or more counters from one single pile
- The winner is the player to take the last counter from the table.

The game can be played physically with counters on a central table or an overhead projector, or virtually using a slide in a PowerPoint® presentation and/or an interactive whiteboard. The counters can be replaced by pencils, rubbers, squares of card, matchsticks, marbles etc.

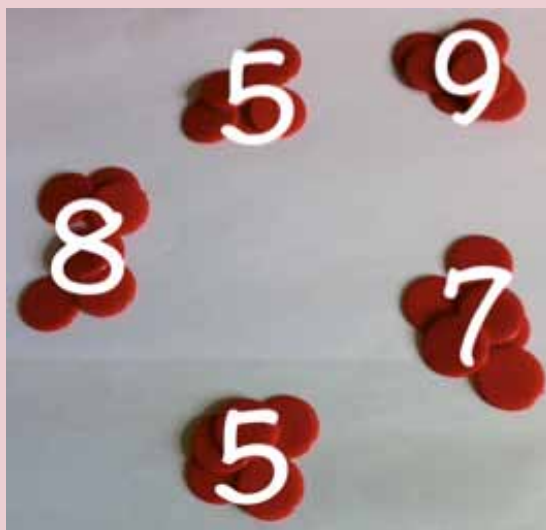
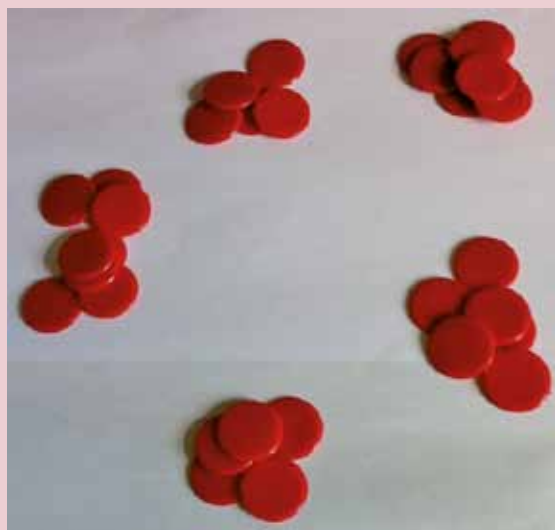
Challenge the class, saying 'teacher always wins', and play the game.

How to always win

Look at the piles like a computer, counting in binary.

After your turn you have to leave the piles so that there are an even number of each of the binary numbers 8, 4, 2 and 1 on the table.

In the images below the total number of counters can be divided into two '8's, three '4's, one '2' and four '1's:



Number of counters in the pile	8	5	9	7	5
Binary	8421 1000	8421 0101	8421 1001	8421 0111	8421 0101

You must remove a '4' and a '2' to make an even number of the binary numbers, so take six counters from the pile containing seven counters. If you leave the table with only even numbers of the binary numbers, you will always win!

Before the lesson, try playing the game against someone who does not know the winning algorithm.

When you have shown the class that you can always win, let them play in pairs.

While the students are playing, go to one table and teach the students the 'trick'. When you are confident that they can play and always win then announce to the class that you have some 'champions' who will always win. Let them play against others and then get them to explain the trick. Remind the class by explanation and visuals around the room, the structure of binary numbers 1, 2, 4, 8, 16 etc. Give less confident students a copy of the empty table to complete so that they can more easily see which counters to remove:

Number of counters in the pile					
Binary	8421	8421	8421	8421	8421

For less able students, start with just three piles of counters.

For more information about the game of NIM, see www.pgce.soton.ac.uk/IT/Curriculum/Resources/Nim.

Activity 3: Using Scratch to create an interactive game

Scratch is a well-regarded graphical programming development environment, ideally suited to Key Stage 2 and introducing programming in Key Stage 3. There are many useful resources available online and the Computing At School community is highly supportive of the use of Scratch.

Demonstrate the Pong game to the students scratch.mit.edu/projects/10000036:



Then tell the students how to code it by loading the graphics – the paddle, ball and stage – and then dragging blocks from the middle to the right. Remind students of the need for precision when entering code and the value of asking colleagues to debug their coding if it does not work.



Celebrate the success of those students who created the working game and give students a choice of other programs to copy.

Resources and courses

The following is a list of organisations, known at time of going to press, that provide free and paid-for resources and courses that will help you develop and deliver a broad, balanced and integrated curriculum to your students. The fact an organisation is not on this list in no way suggests anything about the quality of the resources they provide. We strongly recommend consulting the CAS forum for a more comprehensive and up-to-date guide to currently well-regarded resources.

MOOCs (Massive Open Online Courses)

Cambridge GCSE Computing Online MOOC

This MOOC has been created by the Cambridge-based partnership of the exam board OCR, Cambridge University Press (CUP) and the Raspberry Pi Foundation. The course is based on OCR's GCSE Computing curriculum and gives participants an excellent opportunity to investigate how computers work, how they are used, and how to develop computer programming and problem-solving skills. The course has been designed for 14- to 16-year-olds, but is free and open to all and can be used either as a course that teachers can sign their whole class up to or a resource to support teachers. Find out more at: www.cambridgegcsecomputing.org.

Computing for Teachers MOOC

This MOOC is run by the Department of Computer Science at Warwick University. It is aimed at teachers preparing to deliver the new computing curriculum, providing the necessary subject knowledge and programming skills required to teach computing confidently at Key Stage 3 and GCSE. The course provides in-depth coverage across three areas: computing concepts, programming in Python and how to teach the concepts. The course has received support from Google, BCS and Computing At School. Find out more at www2.warwick.ac.uk/fac/sci/dcs/schools/cpd.

Teaching Computing MOOC

This MOOC, which is run by the University of East Anglia, is in two parts and aims to prepare teachers to deliver the new curriculum effectively to children in years 5, 6, 7 and 8 through a mixture of subject knowledge and pedagogical advice. It will be valuable for both ICT specialists and primary teaching non-specialists. Expert 'Master Teachers' from Computing At School have designed the course to make sure that teachers have the most up-to-date information that they can take into their classrooms to teach great computing lessons. Find out more at www.uea.ac.uk/study/short-courses/online-learning.

Partner organisations



AppInventor.org motivates students to learn coding and computer science by teaching them to program the devices they carry around with them every day.

The site is based on the App Inventor visual language, which allows you to code apps by plugging together high-level puzzle pieces. AppInventor.org offers video and text lessons for self-directed students and a course-in-a-box (www.appinventor.org/course-in-a-box2) that has been used as a model for numerous primary school, secondary school and university level courses.



AppShed Academy (appshed.com/academy/appshed-academy) is a free learning resource available to everyone using AppShed. It covers the basics of app creation right through to advanced programming techniques. Lesson plans are available for teachers, whilst students can use the self-paced, step-by-step video tutorials. Image packs, sample code, course notes and videos can be downloaded and used offline. Teachers are able to track students' progress, monitoring and viewing students' apps and controlling the distribution of apps. Students can also provide self-assessment feedback allowing teachers to monitor their proficiency with course materials. This greatly enhances peer-to-peer learning and the ability for teachers to identify areas of weakness.

BBC

BBC Bitesize (www.bbc.co.uk/education) is a free online resource for students and teachers, and includes a range of new content to specifically support the new computing programme of study (www.bbc.co.uk/education/subjects/zvc9q6f).

Megabits (www.bbc.co.uk/programmes/b01kl16t/clips) are a series of short videos that give students an insight into how computers actually work. Filmed in real-life work settings, the videos look closely at what a computer consists of, how the various components work, how it processes data, and how it is used in robotics and software development. And in **Cracking the Code** (www.bbc.co.uk/programmes/b01r9tww/clips), Minna Kane and her team of young hackers explore the world of computer programming in a special compilation of short films for primary aged children.

BCS

BCS is the professional body for IT and is governed by Royal Charter, which includes a primary goal of advancing computing education for the benefit of the public. BCS also promotes wider social and economic progress through the advancement of information technology science and practice. BCS accredits computing degree courses in over 90 universities around the UK.



CAS (www.computingatschool.org.uk) is a grass roots, school-led, organisation and its energy, creativity and motivating force comes from its members. CAS is formally part of BCS - The Chartered Institute for IT.

CAS aims to inspire, equip and support schools and teachers to deliver the computing curriculum with confidence and enthusiasm. It currently has over 15,000 members, of whom about 75% are school teachers, but also includes many computing professionals from global IT companies and academics from world-leading universities. CAS runs over 110 local hubs around the country to nurture peer-to-peer professional communities of practice, many of which are run with support from local universities.

CAS also runs the Barefoot Computing project (<http://barefootcas.org.uk>), which is funded by the DfE. This project is producing exemplary teaching resources for primary schools, to illustrate how teaching computer science can also improve learning in English, maths, science and history within a cross-curricular environment. These resources may also be of interest to secondary school teachers keen to learn what is happening in primary schools or to use as inspiration.



Codecademy is a free online interactive platform for learning programming languages. Founded in the US, it is committed to creating the best possible learning experience for teachers and students. It has seen its user base skyrocket to over 25 million and has now developed free resources to help teachers in the UK prepare for the new computing curriculum. It offers:

1. **Teacher training:** The online courses help teachers learn the fundamentals of programming across multiple languages, including JavaScript, Python and PHP.
2. **Class resources:** The online courses are supported with schemes of work, lesson plans and quizzes, all mapped to the National Curriculum Computing Programmes of Study.

3. Pupil Tracker: Enables teachers to create bulk accounts for students and track individual and class progress through the courses.

Find more at www.codecademy.com/schools/curriculum / @CodecademyTeach.



cs4fn (Computer Science For Fun, funded by the EPSRC with support from Google, www.cs4fn.org) is a national public engagement project from Queen Mary University of London to enthuse school students about computer science. It sends free fun magazines on interdisciplinary computing research, aimed at young people, to schools and home educators. cs4fn also has funding, from the Mayor of London and the Department for Education, to work with King's College London on a sister project, Teaching London Computing (teachinglondoncomputing.org), supporting computer science teachers to deliver the new computing curriculum. It provides CPD courses for teachers, free workshops and free resources to download. The website is full of fun 'unplugged'-style activities to try out in class or at home.



The CS Unplugged project (csunplugged.org) is a free source of ideas for computing activities away from the computer. The activities teach fundamental ideas from computer science using games, magic tricks and puzzles to engage students. Many involve physically running around, which is a great break from online activities. The original Unplugged work was aimed at primary school students, but is widely used in high schools and universities to introduce topics that might sound difficult but can be taught easily using the scaffolding provided (e.g. binary numbers, data compression, searching and sorting algorithms). There is a sister project aimed at high school students, which is an online, open source textbook, called the *Computer Science Field Guide* (csfieldguide.org.nz). This covers topics in a little more depth, and includes videos and online interactive activities. It currently matches the New Zealand computer science curriculum requirements, but will be expanded to cover topics needed in other curricula.



The Digital Schoolhouse is a primary to secondary transition project focused on delivering inspirational computing. Ukie, the trade body for the Interactive Entertainment industry, and the Digital Schoolhouse Trust have established Digital Schoolhouses in ten London secondary schools. Each Digital Schoolhouse aims to grow and support a network of primary teachers to deliver creative and cross-curricular lessons with computing at their heart and offers enrichment days for primary school pupils. Inspirational Key Stage 2 and 3 resources are also freely available for download from the website: www.digitalschoolhouse.org.uk.

Greenfoot



Greenfoot is an educational programming environment aimed at beginners in programming aged 13 to 20. It uses Java as its programming language and is free and open source.

Greenfoot combines a standard programming language – Java – with an engaging interactive environment that makes it very easy to create animated graphical applications. Learners typically develop their first small program within the first half hour, and quickly learn to program simple games and simulations. While the attraction for many learners is the playful, game-like context, Greenfoot is carefully designed to teach fundamental object-oriented programming constructs and concepts that are then directly transferrable to other environments. It is an ideal successor system after block-based languages, such as Scratch.

Many teaching and learning resources are available, both for students and for teachers. Learners find examples and a community at www.greenfoot.org. Excellent support for teachers and extensive teaching material is available at <http://greenroom.greenfoot.org/door>.



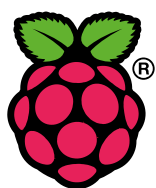
Kodu lets children create games on the PC and Xbox via a simple visual programming language. Kodu can be used to teach creativity, problem solving, storytelling and programming. Anyone can use Kodu to make a game, young children as well as adults with no design or programming skills. Kodu on the PC is a free download and runs on Windows XP, Vista, Win 7 and Win 8 and currently supports 17 languages. For more information, visit www.kodugamelab.com.

Python School

Python School (pythonschool.net) is a website containing many video tutorials and exercises to help teachers learning to teach Python programming in school, from the basics right up to A Level standard. The materials for beginners include how to use selection, iteration and assignment statements in Python. As you progress, there are tutorials on using Python with databases, with server-side applications, using regular expressions and building a complete application to the standard of an A Level project using PyQt. The Python School materials were developed in 2011 to support face-to-face CPD sessions and are used by thousands of teachers each month.

Raspberry Pi

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV and uses a standard keyboard and mouse. It enables people of all ages to explore computing and to learn how to program in languages such as Scratch and Python. It is capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing and playing games. It is also able to interact with the outside world and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras. For more information, go to www.raspberrypi.org.



Raspberry Pi is a trademark of the Raspberry Pi Foundation.



RoboMind Academy (www.robomindacademy.com) trains Computational Thinking: an essential 21st century skill.

By programming a virtual robot in the e-Learning environment, the student is introduced to logic, automation and technology. Logical thinking is directly connected to solving real world challenges. RoboMind is an effective and motivating introduction that is used worldwide with students from age 9. RoboMind Academy makes it easy for teachers. It comes with complete interactive lessons. Courses can be completed at your own pace, thanks to automatic guidance by the virtual mentor. That makes it feasible to lecture large groups of students at the same time. As a teacher you can follow the progress of students at a glance and even share their results in the form of a programming competition on a Smartboard in the classroom. RoboMind works everywhere: from tablets to PCs, online or offline, at home or at school.



ScratchEd (scratched.gse.harvard.edu) is an online community for teachers interested in, or already actively working with, the Scratch authoring environment. With ScratchEd, educators can share stories, exchange resources, ask and answer questions, and find other educators. Since ScratchEd's launch in 2009, more than 13,000 teachers have joined the community, and have shared hundreds of stories and resources.



Delivered across Wales, Technocamps (which was established in 2003 as a schools outreach programme in the Computer Science Department at Swansea University) provides hands-

on practical workshops to inspire, motivate and engage people with computational thinking; and to promote computer science as underpinning all

aspects of modern society. All of the resources for our workshops are freely available online at www.technocamps.com.

As part of the Technocamps programme, we offer Primary school engagement through our Playground Computing outreach project, which is designed to teach the fundamentals of computer science to primary pupils, typically without using computers.

Through our Technoteach project we provide Continuous Professional Development (CPD) for Primary and Secondary school teachers to up-skill them in computing and technology and provide guidance on how to apply these skills in the classroom.



What is it?

TouchDevelop (www.touchdevelop.com) is a browser-based, cross-platform development environment. Regardless of the device and operating system you and your students have – iOS, Android, Apple, Linux or Windows – you can use TouchDevelop. It can be used by absolute beginners, supported by fully-guided built-in tutorials, is touchscreen-friendly and has keyboard and gamepad functionality. There are no downloads to install, yet an offline feature is available. This makes it a comprehensive development and coding environment for schools.

TouchDevelop has a simplicity and yet a richness that makes programming exciting, challenging and rewarding. Code can be written directly on any device. You can utilise the device's sensors, like Bluetooth, GPS and accelerometers, or media, including mp3 and video, using high-level APIs. It is easy to create your own apps and to publish them or tweak those published by others. The concepts you learn readily transfer to traditional programming languages such as Java or C#.

Gamification approach to learning

TouchDevelop encourages users to develop their own code and become better developers through a game-centered approach. When a user creates an account their scripts are saved and can be edited and shared. Users can earn points by completing coding activities or tutorials and can be given points by other users. Users have access to a forum where they can exchange ideas and ask for help from the TouchDevelop community. This community element is continuously evolving. Currently in beta, TouchDevelop

will allow users to work collaboratively on the same script remotely, opening up fantastic opportunities for students and schools to develop projects globally.

Creating apps

TouchDevelop embraces the 'Bring Your Own Device' revolution by providing a unified programming environment everywhere. TouchDevelop does not confine you to one platform or store. For example, it is possible to use an iPad to develop a Google app. Signing up for the DreamSpark programme (www.dreamspark.com), which is free to schools who join CAS, will give your students all the high level tools they need to turn their code into an app and a free Windows Store account to publish it. The extensive library of APIs allows students to incorporate cloud services into their apps, giving them a real-world experience of app development.

Connecting to external hardware

TouchDevelop is a great platform to develop for hardware. APIs are available that allow students to design apps that interact with the sensors on their device and, on a simpler level, TouchDevelop enables code to incorporate a gamepad or a keyboard. Commands also link directly to other hardware development platforms. Makey Makey, Lego® Mindstorms and Arduino Esplora can be coded through TouchDevelop, and more hardware will be added in the near future.

Courses and learning

TouchDevelop has an extensive range of tutorials, support materials and courses, ranging from beginners to advanced, available for free. Many have been created by teachers and there is a strong network of support and advice. 'TouchDevelop scheme of work' on the CD-ROM provides a taster of the materials available. You can also build your own tutorials to guide students through completing an activity in TouchDevelop.

Vendor qualifications



Microsoft® IT Academy Program

The Microsoft IT Academy program (www.microsoftitacademy.com) provides resources, including e-learning, digital academic courseware,

lesson plans and teacher certifications, to support the teaching and learning of the latest technology skills.

Within the Microsoft IT Academy program, Microsoft certifications help bridge the gap between the classroom and the workplace, validating skills and knowledge, enriching students' learning experiences and supporting teachers to develop their skills:

- The Microsoft Office Specialist (MOS) (<https://www.microsoft.com/learning/en-gb/mos-certification.aspx>) certifications help build skills across a number of different Microsoft Office applications, including Word, PowerPoint, Excel, OneNote and Office365. Teachers get ten MOS vouchers per year with IT Academy program membership.
- The Microsoft Technology Associate (MTA) (<https://www.microsoft.com/learning/en-gb/mta-certification.aspx>) certifications cover foundational technology skills for both developer and IT infrastructure. Microsoft have mapped their MTA certifications to the GCSE specifications offered by AQA, OCR, WJEC and Edexcel, so you can offer your students the opportunity to gain national recognised professional qualifications alongside their GCSE. Teachers get ten MTA vouchers per year with IT Academy program membership. Prodigy Learning are also offering a free MTA teacher training bundle, worth £164, up until 30 June 2015. This includes MTA online video training, an MTA practice test and an MTA exam voucher. You can find out more here: www.prodigylearning.com/msukteachertraining.

It also provides professional development for educators:

- The Microsoft Certified Educator (MCE) (<https://www.microsoft.com/learning/en-gb/mce-certification.aspx>) certification validates that teachers have achieved the global educator technology literacy competencies needed to provide a rich learning experience for their students, aligned to UNESCO standards. Teachers get ten MCE vouchers per year with IT Academy program membership.

The Microsoft IT Academy Certification Roadmap, which can be found on the CD-ROM, shows you how to choose the right certifications for you and your students.

Microsoft Virtual Academy (MVA) (www.microsoftvirtualacademy.com) offers hundreds of online Microsoft training courses, delivered by experts, for free. Some of these courses also support the MTA certifications and beginners in technology, and provide teachers with an excellent opportunity to fill gaps in their knowledge base.

The mapping documents are on the CD-ROM:

- Microsoft mapping document for AQA GCSE
- Microsoft mapping document for OCR GCSE
- Microsoft mapping document for WJEC GCSE
- Microsoft mapping document for Edexcel GCSE

Cisco Networking Academy and Oracle University

Cisco and Oracle also provide professional vendor certifications, which students could work towards alongside their academic qualifications. More information can be found at www.netacad.com and education.oracle.com.

Recommended books about computing

Members of the CAS community have compiled a list of inspirational yet accessible books about computer science: community.computingschool.org.uk/resources/199.

And a list of books that are recommended for secondary PGCE students: community.computingschool.org.uk/resources/1787.

Also recommended are:

- Bird, J., Caldwell, H. and Mayne, P. (eds). *Lessons in Teaching Computing in Primary Schools* (Learning Matters, 2014).
- Williams, L. (ed.). *Introducing Computing: A Guide for Teachers* (Routledge, 2014).
- Hey, T. and Pápay G. *The Computing Universe: A Journey Through a Revolution* (Cambridge University Press, 2014).
- Simons, C. and Hawkins, C. *Teaching Computing*, 2nd edn (Sage, forthcoming May 2015).

The logo for Cambridge University Press, featuring the word "CAMBRIDGE" in a white, serif, all-caps font centered within a solid black rectangular background.

Available from spring 2015, the brand new A/AS Level Computer Science resources from Cambridge University Press are written specifically for the OCR and WJEC/Eduqas 2015 specifications. They equip students with the skills and enthusiasm necessary to apply their computing knowledge in the real world, whilst helping prepare them for Higher Education and beyond. The resources:

- Are written by an author team of practising teachers including Computing At School master teachers

- Have a strong focus on independent learning, computational thinking, programming and problem-solving skills
- Include rich digital assets showcasing the relevance of computer science to the real world
- Support the transition from ICT to Computer Science
- Are differentiated to support all abilities.

For more information, go to: www.cambridge.org/ukschools.



We are a publishing and training company specialising in products to support the teaching of computer science in schools and colleges. Incorporated in 1999, we have brought a range of products and services to the market to support the teaching of A/AS Computing/Computer Science.

New Publications for 2015:

- AQA A Level Computer Science Unit 1
- AQA A Level Computer Science Unit 2
- AQA AS Level Computer Science Units 1 & 2
- How to Program Effectively in Delphi for AS/A Level Computer Science

New Publications for 2016:

- GCSE Computer Science
- How to Program Effectively in C# for AS/A Level Computer Science

For more information, please go to www.educational-computing.co.uk.



Hodder Education publishes blended print and digital learning resources to support the teaching and learning of computing progressively from Key Stage 3 through GCSE to A level. Key publishing for each range includes:

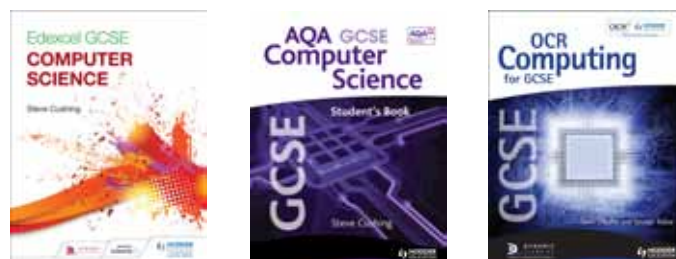
Compute-IT for Key Stage 3

Student's books, teaching packs and a suite of digital teaching and learning resources, delivered through Dynamic Learning, for the new KS3 curriculum. Compute-IT CPD Video Lessons feature over 150 tutorials to support the delivery of key topics in the new Programme of Study.



For more information go to www.hoddereducation.co.uk/compute-it.

AQA, Edexcel and OCR Computer Science for GCSE



For more information go to www.hoddereducation.co.uk/computing.

AQA and OCR Computer Science for the new A level curriculum



For more information go to www.hoddereducation.co.uk/Alevelcomputing.



PG Online produces a complete and comprehensive series of KS3 teaching materials to help support the national demand for resources for the new computing curriculum. The downloadable six-week units each include editable lesson plans, PowerPoint presentations and worksheets and are sold as a lifetime site licence.

They have proved to be a phenomenal hit with teachers nationwide, and especially with those who may be inexperienced in teaching computing rather than IT.

Two exciting new series of downloadable teaching units are now available for GCSE OCR Computing and AQA Computer Science. To accompany these units, PG Online has published new editions of Susan Robson's popular GCSE textbooks, which are also available as downloadable PDF versions.



See www.pgonline.co.uk for more details.

RISE★STARS

Educational Publishing Specialists

To get you started if students haven't done any computing at primary school:

Switched on Computing

This scheme for primary schools is published by Rising Stars in association with Computing At School and NAACE. Full details can be found at www.switchedoncomputing.co.uk.



Switched on Computing – take your first easy steps with Microsoft

This free resource has six project ideas each using Microsoft tools. Schools can download this resource at www.switchedoncomputing.co.uk/microsoft. Supporting videos for this resource can also be found here: raychambers.wordpress.com/2014/09/20/switched-on-computing-tutorial-samples.



OXFORD

UNIVERSITY PRESS

Oxford University Press publishes student books and online resources for computing at primary, GCSE and A Level.

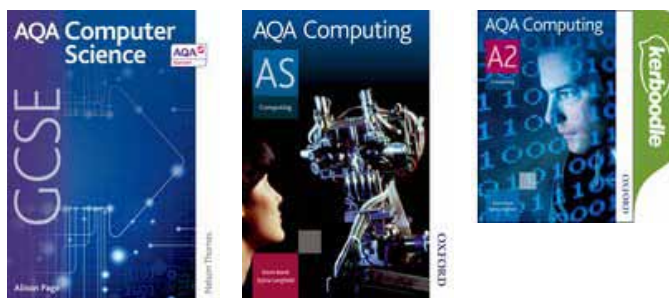
Oxford International Primary Computing



For more information, go to: <https://global.oup.com/education/content/primary/series/computing/?region=international>.

AQA GCSE and A Level Computer Science

Our AQA GCSE and A level resources have been written by leading authors to support the latest specifications, focusing on developing programming knowledge, computational thinking and problem-solving skills. AQA A Level Computing is also accompanied by Kerboodle, which includes a wide variety of classroom resources, time-saving lesson presentations and assessment tasks to help track learning.



For more information, go to: <https://global.oup.com/education/secondary/subjects/ict/?region=international>.

Other sources of inspiration

- STEM Ambassadors: www.stemnet.org.uk/topboxes/stem-ambassadors.
- Code clubs: www.codeclub.org.uk.









Section 6: Assessment and progression

How do I assess progress?

The system of levels is being abolished. How your school chooses to assess, record and report progression is consequently entirely up to you. You could use the Progression Pathways as a guide to what progression might look like. The *What are the Progression Pathways?* information sheet explains why there are two versions of the Progression Pathways and describes how they can be used to help you assess students' progress. Both versions of the Progression Pathways are provided for you to use. The *Interactive Progression Pathways Tool* – which illustrates the links between the Progression Pathways, the National Curriculum Programmes of Study for Computing and the Computational Thinking Framework – will also be useful in helping you evidence and assess progression. Links to guidance on assessment from September 2014 from the Department for Education, Ofsted and NAHT are also provided.

The *What different types of assessment are there?* information sheet discusses both formative and summative assessment. Pages 25 and 26 of *Computing in the National Curriculum: a guide for secondary teachers* also provides information about formative and summative assessment.

Finally, in this section, *How do I assess programming?* is a practical activity designed to illustrate how to assess students' programming skills.

		Book	CD-ROM	Website
	Information sheet: <i>What are the Progression Pathways?</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.	✓ Pages 12–13	✓	✓
	<i>Progression Pathways: topics</i> © Mark Dorling and Mathew Walker. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.	✓ Pages 14–15	✓	✓
	<i>Progression Pathways: strands</i> © Mark Dorling and Mathew Walker. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.	✓ Pages 16–17	✓	✓
	<i>Interactive: Interactive Progression Pathways Tool</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.		✓	✓
	Links to: Guidance on assessment from Department for Education Guidance on assessment from Ofsted Guidance on assessment from NAHT		✓	✓
	Information sheet: <i>What different types of assessment are there?</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.	✓ Page 54	✓	✓
	Information sheet: <i>Computing in the National Curriculum: a guide for secondary teachers</i> © Computing At School 2013 This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.		✓	✓
	Activity: <i>How do I assess programming?</i> © Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.	✓ Pages 55–60	✓	✓

What different types of assessment are there?

Two categories of assessment need to be considered for students at Key Stages 3 and 4. Both formative assessment and summative assessment are required in the classroom, but serve two different purposes. Summative assessment is familiar to teachers as unit tests or public examinations. It takes place after the learning phase and is used to quantify achievement. On the other hand, formative assessment is a continual process, whose purpose is not to quantify, but to influence improvement. This is familiar to teachers in various forms such as peer assessment, marking of drafts, and questioning. It incorporates some type of feedback provided for the student. If actioned, the feedback should afford improvement in the student's understanding or output.

It is possible to use summative assessment to assess skills and knowledge. Marking a worksheet showing the completion of different sorting algorithms or a multiple-choice test of terminology are valid ways of testing knowledge. Checking the use of formulas in a spreadsheet or the accuracy of a flowchart are valid ways of testing skills. Summative assessment of artefacts can only report on outcomes, not on the processes employed to reach those outcomes.

The processes used to produce many of the outcomes in computing are referred to as computational thinking. Computational thinking concepts include decomposition, abstraction, algorithmic thinking, generalisation, and evaluation. Pattern recognition is also often viewed as a component of computational thinking within the classification of generalisation. The outcomes produced by learners may not directly afford an opportunity to assess the use of these particular thinking skills. For example, a learner designs an algorithm to change a bicycle tyre. The steps are ordered correctly to provide a solution. Based on the artefact presented, it is tempting to attribute the computational thinking process of decomposition (steps), algorithm design (ordered), and evaluation (works) to the learner. However, the decomposition (steps) could have been provided by the teacher and the evaluation (works) could have been determined by a peer. The ability to assess

computational thinking is intrinsically tied to the techniques used in the classroom. In this particular instance, only the algorithm design (ordered) is verifiably the learner's own behaviour. This is why it is more challenging to assess the thinking than it is to assess skills and knowledge.

In reality, assessing computational thinking could be incorporated into the formative assessment stage. Shifting the focus from the endpoint to the development provides opportunities for learners to demonstrate their use of computational thinking skills such as decomposition and abstraction. Providing a range of different problem contexts gives opportunities for learners to demonstrate generalisation of problem solutions. Helping a learner debug a program gives teachers opportunities to observe the use of evaluation and algorithm design. Questioning gives learners the opportunity to demonstrate their analytic skills and gives teachers the opportunity to observe the use of a variety of computational thinking skills. In these ways, assessing computational thinking may be accomplished in situations where formative assessment is being used.

In terms of assessment, there are several needs that require balancing: the need for summative assessments to provide data; the need for formative assessments to afford progression; and the need for assessing knowledge, skills, and computational thinking as required in the programme of study. By focusing on the use of formative assessment during learning, it is possible for teachers to observe the use of computational thinking skills without having to infer their use from the characteristics of an artefact. In that way, formative assessment becomes just as valuable for the teacher as for the learner. Summative assessment, when required, then validates the good foundations of formative assessment.

The recording of learning outcomes is done in recognition of the importance in monitoring progression over a sustained period of time. In order to effectively record progression in computing, the computational thinking concepts used by the students and the classroom techniques used by the teacher to facilitate learning require periodic auditing.

How do I assess programming?

Programming isn't difficult if you know how to solve problems. If you take the same approach to assessment, it is just as easy. Use the guidance provided below to assess each of the programs provided.

Assessment criteria

The assessment criteria associated with judging a computer program have five simple aspects:

1 Does the program run?

This is not necessarily the most important aspect. A very good program with all the features of readability, understandability, extensibility and general value as a product, might not, at the point of judgement, work. When learning to program (and as an experienced programmer developing a product) much of the time, the program does not work!

However, even though a good program may not run, it should be clear that it can be made to work with only minor modifications.

2 Is the program formatted for readability?

When looking at the code with a squinted eye, does it look formatted with indentations and blank lines to indicate the programming structure of loops, procedures, modules etc?

```
110 REM Nested Loops
120 FOR n1 = 1 TO 10
130   PRINT ""
140   PRINT n1;
150   PRINT " men went to mow, went to mow a meadow"
160   FOR n2 = n1 TO 1 STEP-1
170     PRINT n2;
180     PRINT " men"
190   NEXT n2
200   PRINT "and his dog, went to mow a meadow"
210 NEXT n1
220 END
```

3 Is the program 'commented' for understandability?

Within the code are there comments that indicate what the program is doing at that particular point?

Are the contents of variables described at the point when they are first used or changed? Is the purpose described at the start of the procedure and when it is called? Are the names of the programs, variables and procedures indicative of their purpose?

In BASIC and Visual BASIC a comment is a REM statement, in Python #, in Java and MySQL // or /* */; in Scratch comments can be added or a block created and inserted.



4 Is the program extensible and portable?

Can the code created by the student be used in other contexts? Do they understand how the code could be adapted for other uses, either in the same program or in other applications? Has the code the potential to be used by others in the development of other programs? This is a high level skill which requires the student to be aware of the readability and understandability issues as well as the potential for the code to be ported into other languages/ computers.

5 Is the product good?

This aspect of the quality of code is the most generic and overarching. It is about making a judgement about the 'product'; that is, the user experience of the program. Judgements can be made by asking whether it meets the criteria (original specifications for the design or regulations/rules/requirements associated with the product) and the heuristics (the general 'rules of thumb') associated with the specific area of programming. How effective is the program, how well does the program meet the requirements, and how efficient is the program, in terms of coding time, running time and resources?

An important observation: the programming language and the programming environment will determine whether certain aspects can be or must be implemented.

Program 1: 'One man went to mow' in Scratch

Does the program run?	
Is the program formatted for readability?	
Is the program 'commented' for understandability?	
Is the program extensible and portable?	
Is the product good?	

Program 2: 'One man went to mow' in BASIC

Does the program run?	
Is the program formatted for readability?	
Is the program 'commented' for understandability?	
Is the program extensible and portable?	
Is the product good?	

Program 3: 'Sine wave' in BASIC

Does the program run?	
Is the program formatted for readability?	
Is the program 'commented' for understandability?	
Is the program extensible and portable?	
Is the product good?	

Program 4: 'Calendar' in pseudocode

Does the program run?	
Is the program formatted for readability?	
Is the program 'commented' for understandability?	
Is the program extensible and portable?	
Is the product good?	

Program 1

Task: Using the online version of Scratch, create a program that prints the grammatically correct words for the song 'One man went to mow'.

To test this program go to scratch.mit.edu/projects/25828736.



Program 2

Task: In BASIC, create a program including iteration and selection that prints the grammatically correct words for the song 'One man went to mow'.

```

110 REM Nested Loops
120 FOR man = 1 TO 10
130 PRINT ""
140 PRINT man;
150 IF man = 1 THEN PRINT " man";
160 IF man > 1 THEN PRINT " men";
170 PRINT " went to mow, went to mow a meadow"
180 FOR countdown = man TO 1 STEP -1
190 PRINT countdown;
200 IF countdown = 1 THEN PRINT " man";
210 IF countdown > 1 THEN PRINT " men"
220 NEXT countdown
230 PRINT " and his dog, went to mow a meadow"
240 NEXT man
250 END
  
```

To test the program, copy the code into this emulator:

- www.calormen.com/jsbasic

Program 3

Task: Write a program that creates a graphical impression of a sine wave.

Hint: SIN(x) produces a value between -1 and +1 for values of x between 0 and 2*PI

```

100 REM X counts the number of steps (*s) to be printed
101 REM Y can calculates the height of the *
102 REM X/5 gives 36 * for every cycle
  
```



```

103 REM Z counts the spaces before the * is displayed
104 REM 22/4/13 version 1.04
110 FOR X = 1 TO 1000
120 LET Y = (SIN(X/5)*20)+30
130 FOR Z = 1 TO Y
140 PRINT " ";
150 NEXT Z
160 PRINT "*"
170 NEXT X
180 END

```

Program 4

Task: Write, in pseudocode, an algorithm for printing out the dates of the year using two arrays:

- One containing the months of the year and the number of days in each month.
DATA "January", 31, "February", 28, "March", 31, "April", 30, "May", 31, "June", 30, "July", 31, "August", 31, "September", 30, "October", 31, "November", 30, "December", 31
- One containing the days of the week
DATA "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"

Declare the arrays:

M\$(12) to store the names of the months (string)
D(12) to store the days in each month (number)
W\$(6) to store the days of the week (string) W\$(0) is Monday.

Read the data:

REPEAT 12 names and number of days
REPEAT 7 days of the week

Variables:

A counts the months 1 to 12
B counts the days from 1 to the number in the month
C counts the days of the year. Set C to be the day of January 1st.

Nested loops:

from 1 to 12 (January to December)
from 1 to 28, 30 or 31 (depending on month)

output the day of the month, the month and the day of the week w\$(C modular 7)

increment C

DATA "January", 31, "February", 28, "March", 31, "April", 30, "May", 31, "June", 30, "July", 31, "August", 31, "September", 30, "October", 31, "November", 30, "December", 31

DATA "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"

Example judgements using the five aspects

Formative assessment has an important impact on learning. The example judgements on the programs provided below are expressed as guidance for students. Did you make similar points when assessing the programs?

Program 1: 'One man went to mow' in Scratch

This program runs and produces text on the screen with an appropriate and interesting graphic. The colour coding and shapes of the programming structures in Scratch aids readability and the nested loops are clear. The variables have simple, single letter names and there is no indication of what they represent. The graphic helps indicate the context of the application. There is little scope for application in other contexts. The overall value of the program is limited because of the specification and programming environment. It did not meet the specification because the grammar of the output is incorrect. For example, 'One *men* went to mow'. The exercise requires a selection statement so that, if the value is 1 then it prints 'man' and if it is more than 1 it prints 'men'.



The formatting shows the structure of nested loops and the coding runs. You should consider the use of 'if else' statements to prevent the grammar error of '1 men went to mow'.



Program 2: 'One man went to mow' in BASIC

This program runs and produces text on the screen. The code is formatted to aid readability by successive indents for the nested loops. There is little scope to add blank lines but there is only one REM statement to help structure the program. The variables have names that represent their use. There is no description of the process or the context of the application. There is little scope for application in other contexts. The overall value of the program is limited because of the specification and programming environment. It did meet the specification because the grammar of the output is correct, producing 'one man' and 'two men'. The code includes iteration ('for next' loops) and selection ('if then' statements).



The formatting shows the structure of nested loops and the coding runs. The output is grammatically correct. You should consider the use of REM (comment) statements to make the working of your code clear and to help others or yourself modify it later.

Program 3: 'Sine wave' in BASIC

The program runs with no errors. There is no formatting, which suggests the author does not fully understand the significance of nested loops. However, the variables are clearly described and the algorithm indicated through the REM statements. Perhaps the programmer is aware that the application can be used to demonstrate the patterns of cosine and tangent and so there

is potential extensibility. The graphic does not run smoothly, because when the sine value is low there are fewer steps than when the sine value is high, making the speed of rendering variable.



The coding is efficient, using a few well-defined variables and creating an effective visual that meets the requirements. To ensure that the program is readable, formatting should be applied to the code to indicate the nested loop structure.

Program 4: 'Calendar' in pseudocode

This has the potential to run with no error. Conventional syntax is used to indicate variable and array names and structures. The code is sectioned, systematically dealing with initialisation, processes and data. However, there is no explanation of the algorithm and how the day of the week is calculated. An example of output would be useful in helping understand what the program does.



The coding is efficient, using a few well-defined variables and arrays with conventional syntax/ names. The code is clearly structured. You should add more comments to make the algorithm and output more explicit.





Section 7: Next steps

What next?

If you're following our road map for delivering high quality CPD, it will be time to get together again and share your journey with other members of your CAS hub or colleagues in your department. Discuss where you started, where you have been and where you are going now and, if appropriate, update your milestones document.

The *Developing as a teacher of computing* information sheet contains guidance on possible next steps beyond this CPD toolkit.

		Book	CD-ROM	Website
	Planning sheet: <i>Milestones</i> <small>© Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.</small>	✓ Page 27	✓	✓
	Information sheet: <i>Developing as a teacher of computing</i> <small>© Crown Copyright 2015. This content is free to use under the Open Government Licence v3.0.</small>	✓ Pages 62–63	✓	✓



Developing as a teacher of computing

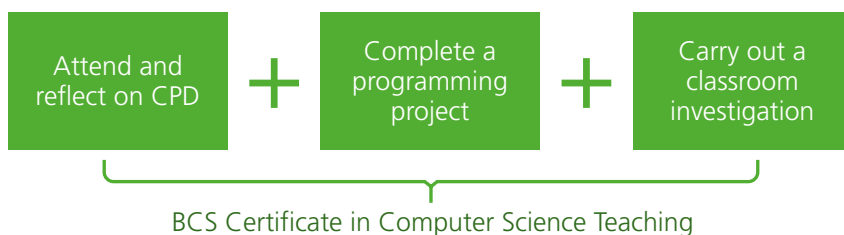
There are a number of ways in which you can develop your expertise as a teacher of computing.

BCS Certificate in Computer Science Teaching

The BCS Certificate provides teachers with professional recognition that they are competent teachers of the computer science elements of computing. It is accredited by BCS, the Chartered Institute for IT.

The BCS Certificate is a roll-on, roll-off programme that teachers can join at any time. It is evidence-based certification, whereby teachers gather evidence from their own teaching and professional development to demonstrate three areas of competence:

- Ability to take responsibility for their ongoing professional development by participating in and reflecting on CPD.
- Technical subject-knowledge skills appropriate to teaching in either primary or secondary schools.
- Ability to develop and extend their pedagogical content skills in computer science in a reflective manner.



Teachers who have completed Microsoft's Software Development Fundamentals MTA (www.microsoft.com/learning/en-us/exam-98-361.aspx) can claim an exemption from the programming project, as can teachers who have a recent computer science-related degree that includes a final-year software project.

The BCS Certificate in CST is open to all qualified teachers who are currently teaching computing.

On enrolment, you will be assigned to an e-assessor and given access to the Certificate Virtual Learning Environment (VLE). The VLE offers guidance on completing the three separate parts of the Certificate and facilitates contact between you and your e-assessor. You will receive feedback on proposals for projects and drafts of your work; the assessment model is based on formative feedback being an important aspect of learning. You have one year to complete and submit your work.

The Certificate is valid for five years and, after this time you can renew your Certificate. The renewal process ensures the currency and credibility of the Certificate as a professional qualification. The Certificate offers you a professional, rather than an academic, accreditation. However, completing it may be a stepping-stone to a Master's degree or another academic qualification.

More information can be found at computingatschool.org.uk/certificate.

Network of Teaching Excellence in Computer Science

You can register your school with the Network of Teaching Excellence in Computer Science (NoE). By doing this, your school is saying, loud and clear, that:

- Computing, as a subject, is important.
- Computing, as a subject, is part of the school development plan.
- Our staff has access to suitable training.

The NoE is coordinated by Computing At School (CAS) working in collaboration with the British Computing Society (BCS) Academy, and is supported by the Department for Education, the awarding bodies, the Council of Professors and Heads of Computing (CPHC), Microsoft and Google. You can find out more information about registering your school with the Network of Excellence at www.computingschool.org.uk/index.php?id=member-schools.

You could also register your school as a Lead School within the Network of Excellence, if you are willing and able to make time to support colleagues in other schools. You can find out about the benefits of being a Lead School at www.computingschool.org.uk/index.php?id=noe-lead-schools.

Master Teacher

CAS Master Teachers deliver local CPD and play a vital role in developing computing provision in schools across the country.

If you can answer 'yes' to the following questions then the Master Teacher programme is for you!

- Are you a primary or secondary teacher in a state maintained school?
- Would you like funding to develop your skills and knowledge of computing in the new National Curriculum?
- Are you an experienced teacher with 'good with outstanding' or 'outstanding' teaching seen in your recent lesson observations?
- Can you confidently engage with your peers in a professional environment?
- Do you have the ability to design and deliver practical and interactive workshops for teachers with appropriate course material?
- Do you have a passion about sharing best practice in teaching and learning?

If you have insufficient subject knowledge to teach computing at the key stage(s) you are trained to teach, then you can become a Level 1 (Trainee) Master Teacher. You can find out more information here: www.computingschool.org.uk/index.php?id=noe-master-teachers-level-one.

If or when you have sufficient subject knowledge and experience then you can become a Level 2 Master Teacher. More information about how to apply can be found here: www.computingschool.org.uk/index.php?id=noe-master-teachers-level-two.

A word about sponsorship. The QuickStart project was funded by Microsoft, with matched funding from the Department for Education, and it is heartening to see such tangible support for teachers from both business and government. Computing At School would like to thank both of them warmly, and emphasise that **QuickStart Computing: a CPD toolkit for secondary teachers** was developed for you by a Computing At School working group, without the direct influence of either sponsor.

Although every effort has been made to ensure that website addresses are correct at time of going to press, Computing At School cannot be held responsible for the content of any website mentioned. It is sometimes possible to find a relocated web page by typing in the address of the home page for a website in the URL window of your browser.

© Crown Copyright 2015

This content is free to use under the Open Government Licence v3.0.

Cover and text design: Burville Riley Partnership
CD-ROM built by Tag Publishing Services

Typeset in Tenso Regular 11/13pt by DC Graphic Design Ltd., Swanley Village, Kent.
Printed and bound in the UK

A catalogue record for this title is available from the British Library.

ISBN: 9781471848070

Acknowledgements

Consultant: Mark Dorling
Developed by: Abigail Woodman

Thank you to Steve Connolly, Anselm Eustace, Deborah Sanderson, David Smith, Debbie Smith and Gavin Summers from Hodder Education, to Teresa Watts and Helen Royle from Stuck Ltd, to Alex Buckley from Rolemap Arts Ltd and to Excelsoft Technologies.

Thank you to the following people who contributed their ideas and support and for granting us permission to include copyright material:

Miles Berry, University of Roehampton, for *Paired programming*.

CSUnplugged for permission to use *Beating the clock - sorting networks* and *The Orange Game*.

Ray Chambers and Roger Davies for reviewing the toolkit as it developed.

The Barefoot Computing project and Jon Chippindall for permission to use the screenshot from 'One man went to mow' in *How do I assess programming?*

Mark Dorling for *How to develop schemes of work*.

Mark Dorling and Pete Marshman for *Developing computational thinking in the classroom*.

Mark Dorling and Dave White for leading the action research described in *From graphical to text-based programming languages* and to Yota Dimitriadis for support in framing the discussion.

Mark Dorling and Matthew Walker for permission to use the Progression Pathways.

Hodder Education for permission to use *Introducing the 2014 National Curriculum, Teaching computing to all* and *Choosing a programming language*.

Sarah Lawrey for permission to use *Creating an animation*.

'natalie', a member of the Scratch community, for permission to use the screenshots from her Pong game in *What makes an effective activity?*

Thomas Ng, West Berkshire School Improvement Team, for help creating the *Skills and knowledge audit*, the *Interactive Progression Pathways tool* and curating the guidance on lesson observation.

Simon Peyton Jones for writing *Computing ≠ coding and programming*.

George Rouse for writing *What prior knowledge do students starting GCSE need?*

Shahneila Saeed and the Digital Schoolhouse for granting us permission to use the Digital Schoolhouse's *Lesson plan template* and *Shapes calculator scheme of work*.

Dr Cynthia Selby for writing *What different types of assessment are there?*

Sue Sentance for writing *High quality CPD* and for contributing to *Developing as a teacher of computing*.

John Woollard (University of Southampton – Education) for writing *What makes an effective activity?* and *How do I assess programming?*

Jeanette Patterson, Ray Chambers, Helen Caldwell, Stuart Davison, Jane Waite and Shahneila Saeed for suggesting resources to include in the *Interactive Progression Pathways tool* and to Shahneila Saeed for reviewing them all.

TEDxExeter for permission to use *Teaching creative computer science*.

And, finally, thank you to Pete Marshman and the awesome students at Park House School, particularly Laurence Bu-Rashid, Sam Coyne, Dulcie Crosby, James Cross, Carys Davies, Holly Donohoe, Gemma Forte, Connor Humphreys, Jack Hygate, Matthew Larby, Natasha Mathias, Amelia McKay, Grace Norton, James Peale, Fay Penlington, Daniel Roch, Kirsty Stephens, Natalie Wilson and Charlie Woodfield, for their roles in *Creative Computing in Action*.

Photo credits

Page 6: © Alex Slobodkin/iStockphoto.com; Page 13: © Brand X/ Photolibrary Group Ltd; Page 21: © C. Sherburne/Photodisc/Getty Images; Page 42: John Woollard; Page 45–51 : All logos and images © relevant organisations; Page 59: © michaeljung - Fotolia.com; Page 60: Two speech bubbles © Victoria Kalinina - Fotolia

Scratch is developed by the Lifelong Kindergarten Group at the MIT Media Lab. See scratch.mit.edu.