MacIntyre Academies

Quest Academy

Upper School Long Term Computing Systems Plans 2022 – 2023

Years 8 - 11

<table>
<tr><th colspan="7">KS3</th></tr>
<tr><td rowspan="3">Year 8 and 9</td><td>Autumn 1</td><td>Autumn 2</td><td>Spring 1</td><td>Spring 2</td><td>Summer 1</td><td>Summer 2</td></tr>
<tr><td>7 weeks</td><td>7 weeks</td><td>7 weeks</td><td>5 weeks</td><td>6 weeks</td><td>8 weeks</td></tr>
<tr><td>Overview:</td><td colspan="3">TOPIC COVERAGE:</td><td colspan="2">Objectives:</td></tr>
<tr><td>Year 8 and 9

What is the aim of this Programme of study?
Subject knowledge
This unit focuses on the following key areas of networks:

• Searching
• Threats
• HTML and CSS
• Representing Data with Images and Sound
• How Computers Work: Demystifying Computation

• Engagement factors.
• Enquiry based learning.
• Cross Curricular (particularly with subjects which encounter information technology, computer hardware and processing and digital communication/safety).
• Pupil Led Learning.
• Developing practical skills.
• Developing problem solving and critical thinking skills.</td><td colspan="3">Autumn
Autumn 1
Developing for the web

National curriculum links

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

Autumn 2
Representations: from clay to silicon

National curriculum links (Computing programmes of study: Key Stage 3)

Understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits

Spring
Spring 1
Mobile app development

National curriculum links:

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

Understand several key algorithms that reflect computational thinking; use logical reasoning to compare the utility of alternative algorithms for the same problem

Spring 2
Design vector graphics

National curriculum links:

Undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users

Summer
Summer 1
Computing systems

Summer 2
Cybersecurity</td><td colspan="2">Autumn
• Recall that a general-purpose computing system is a device for executing programs
• Recall that a program is a sequence of instructions that specify operations that are to be performed on data
• Explain the difference between a general-purpose computing system and a purpose-built device
• Describe the function of the hardware components used in computing systems
• Describe how the hardware components used in computing systems work together in order to execute programs
• Recall that all computing systems, regardless of form, have a similar structure (‘architecture’)
• Analyse how the hardware components used in computing systems work together in order to execute programs
• Define what an operating system is, and recall its role in controlling program execution
• Describe the NOT, AND, and OR logical operators, and how they are used to form logical expressions
• Use logic gates to construct logic circuits, and associate these with logical operators and expressions
• Describe how hardware is built out of increasingly complex logic circuits
• Recall that, since hardware is built out of logic circuits, data and instructions alike need to be represented using binary digits
• Provide broad definitions of ‘artificial intelligence’ and ‘machine learning’
• Identify examples of artificial intelligence and machine learning in the real world
• Describe the steps involved in training machines to perform tasks (gathering data, training, testing)
• Describe how machine learning differs from traditional programming
• Associate the use of artificial intelligence with moral dilemmas</td></tr>
</table>

		National curriculum links (Computing programmes of study: Key Stage 3) Can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation Can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems	National curriculum links Understand a range of ways to use technology safely, respectfully, responsibly, and securely, including protecting their online identity and privacy; recognise inappropriate content, contact, and conduct, and know how to report concerns	• Explain the implications of sharing program code • Describe what HTML is • Use HTML to structure static web pages • Modify HTML tags using inline styling to improve the appearance of web pages • Display images within a web page • Apply HTML tags to construct a web page structure from a provided design • Describe what CSS is • Use CSS to style static web pages • Assess the benefits of using CSS to style pages instead of in-line formatting • Describe what a search engine is • Explain how search engines 'crawl' through the World Wide Web and how they select and rank results • Analyse how search engines select and rank results when searches are made • Use search technologies effectively • Discuss the impact of search technologies and the issues that arise by the way they function and the way they are used • Create hyperlinks to allow users to navigate between multiple web pages • Implement navigation to complete a functioning website. • Complete summative assessment ***Spring*** • Describe what algorithms and programs are and how they differ • Recall that a program written in a programming language needs to be translated in order to be executed by a machine • Write simple Python programs that display messages, assign values to variables, and receive keyboard input • Locate and correct common syntax errors • Describe the semantics of assignment statements • Use simple arithmetic expressions in assignment statements to calculate values • Receive input from the keyboard and convert it to a numerical value • Use relational operators to form logical expressions • Use binary selection (if, else statements) to control the flow of program execution • Generate and use random integers

				• Use multi-branch selection (if, elif, else statements) to control the flow of program execution • Describe how iteration (while statements) controls the flow of program execution • Use iteration (while loops) to control the flow of program execution • Use variables as counters in iterative programs • Combine iteration and selection to control the flow of program execution • Use Boolean variables as flags • Draw basic shapes (rectangle, ellipse, polygon, star) with different properties (fill and stroke, shape-specific attributes) • Manipulate individual objects (select, move, resize, rotate, duplicate, flip, z-order) • Manipulate groups of objects (select, group/ungroup, align, distribute) • Combine paths by applying operations (union, difference, intersection) • Convert objects to paths • Draw paths • Edit path nodes • Combine multiple tools and techniques to create a vector graphic design • Explain what vector graphics are • Provide examples where using vector graphics would be appropriate • Peer assess another pair's project work • Improve your own project work based on feedback • Complete a summative assessment ***Summer*** • Explain the difference between data and information • Critique online services in relation to data privacy • Identify what happens to data entered online • Explain the need for the Data Protection Act • Recognise how human errors pose security risks to data • Implement strategies to minimise the risk of data being compromised through human error • Define hacking in the context of cyber security • Explain how a DDoS attack can impact users of online services

				• Identify strategies to reduce the chance of a brute force attack being successful • Explain the need for the Computer Misuse Act • List the common malware threats • Examine how different types of malware causes problems for computer systems • Question how malicious bots can have an impact on societal issues • Compare security threats against probability and the potential impact to organisations • Explain how networks can be protected from common security threats • Identify the most effective methods to prevent cyberattacks

<table>
<tr><th colspan="7">KS3</th></tr>
<tr><td rowspan="4">Year 10 and 11</td><td>Autumn 1</td><td>Autumn 2</td><td>Spring 1</td><td>Spring 2</td><td>Summer 1</td><td>Summer 2</td></tr>
<tr><td>7 weeks</td><td>7 weeks</td><td>7 weeks</td><td>5 weeks</td><td>6 weeks</td><td>8 weeks</td></tr>
<tr><td>Overview:</td><td colspan="3">TOPIC COVERAGE:</td><td colspan="2">Objectives:</td></tr>
<tr><td>Year 9

What is the aim of this Programme of study?
Skills focus: Skills focus: Programming techniques and computational thinking

CR2: The course provides opportunities to develop student understanding of the required content outlined in each of the big ideas described in the AP Course and Exam Description (CED).

CR3: The course provides opportunities to develop understanding of the big</td><td colspan="3">Autumn
Autumn 1 | Autumn 2
Sphero- Course 3 (Theme: Brain Breakers)

<u>Design & Development:</u>
The activities involved in planning, creating and evaluating computing artefacts

<u>Programming languages:</u> Sphero Draw/Blocks/Text (based on Java Script)

Makecode Arcade [Intermediate]: Functions, extensions, animation, difficulty levels, multi-player, tile maps

<u>Algorithms:</u>
Being able to comprehend, design, create and evaluate algorithms

<u>Programming languages:</u>
Creating software to allow computers to solve problems

<u>Programming languages:</u> Microsoft MakeCode (Block/Python/Java)

Spring
Spring 1 | Spring 2</td><td colspan="2">Autumn
• can understand and apply the fundamental principles & concepts of computer science.
• practical experience of writing computer programs to solve problems.
• can evaluate and apply information technology, including new or unfamiliar technologies analytically to solve problems
• are responsible, competent, confident and creative users of information and communication technology.
• design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
• use logical reasoning to compare the utility of alternative algorithms for the same problem</td></tr>
</table>

	ideas, as outlined in the Course and Exam Description. CR4: The course provides opportunities for students to develop the skills related to Computational Thinking Practice 1: Computational Solution Design. CR5: The course provides opportunities for students to develop the skills related to Computational Thinking Practice 2: Algorithms and Program Development. CR6: The course provides opportunities for students to develop the skills related to Computational Thinking Practice 3: Abstraction in Program Development. CR7: The course provides opportunities for students to develop the skills related to Computational Thinking Practice 4: Code Analysis. CR8: The course provides opportunities for students to develop the skills related to Computational Thinking Practice 5: Computing Innovations. CR9: The course provides opportunities for students to develop the skills related to Computational Thinking Practice 6: Responsible Computing. CR10: The course provides a minimum of three opportunities for students to investigate different computing innovations.	**Makecode Arcade [Intermediate]: Controls, level design, number generation, dialogue scripts, sprite arrays** <u>Algorithms:</u> Being able to comprehend, design, create and evaluate algorithms <u>Programming languages:</u> Creating software to allow computers to solve problems <u>Programming languages:</u> Microsoft MakeCode (Block/Python/Java).	**Makecode Arcade [Intermediate]: Skills development** <u>Algorithms:</u> Being able to comprehend, design, create and evaluate algorithms <u>Programming languages:</u> Creating software to allow computers to solve problems <u>Programming languages:</u> Microsoft MakeCode (Block/Python/Java)	• use two or more programming languages, at least one of which is textual, to solve a variety of computational problems • understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems • understand how instructions are stored and executed within a computer system • undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users • create, re-use, revise and re-purpose digital artefacts for a given audience, with attention to trustworthiness, design and usability • can understand and apply the fundamental principles & concepts of computer science. • practical experience of writing computer programs to solve problems. • can evaluate and apply information technology, including new or unfamiliar technologies analytically to solve problems • are responsible, competent, confident and creative users of information and communication technology. • design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems • use logical reasoning to compare the utility of alternative algorithms for the same problem • use two or more programming languages, at least one of which is textual, to solve a variety of computational problems • make appropriate use of data structures [for example, lists, tables or arrays]; • understand simple Boolean logic [for example, AND, OR and NOT] and some of its uses in circuits and programming; • understand a range of ways to use technology safely, respectfully, responsibly and securely, including protecting their online identity and privacy; recognise inappropriate content, contact and conduct and know how to report concerns. • undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and
		Summer		
		Summer 1	***Summer 2***	
		Sphero- Course 3 (Theme: Missions) <u>Design & Development:</u> The activities involved in planning, creating and evaluating computing artefacts <u>Programming languages:</u> Sphero Draw/ Blocks/ Text (based on Java Script)	**Sphero- Course 3 (Theme: Navigation)** <u>Design & Development:</u> The activities involved in planning, creating and evaluating computing artefacts <u>Programming languages:</u> Sphero Draw /Blocks/ Text (based on Java Script)	

				analysing data and meeting the needs of known users • create, re-use, revise and re-purpose digital artefacts for a given audience, with attention to trustworthiness, design and usability ***Spring*** ● can understand and apply the fundamental principles & concepts of computer science. ● practical experience of writing computer programs to solve problems. ● can evaluate and apply information technology, including new or unfamiliar technologies analytically to solve problems ● are responsible, competent, confident and creative users of information and communication technology. ● design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems ● use logical reasoning to compare the utility of alternative algorithms for the same problem ● use two or more programming languages, at least one of which is textual, to solve a variety of computational problems ● make appropriate use of data structures [for example, lists, tables or arrays]; ● understand simple Boolean logic [for example, AND, OR and NOT] and some of its uses in circuits and programming; ● understand a range of ways to use technology safely, respectfully, responsibly and securely, including protecting their online identity and privacy; recognise inappropriate content, contact and conduct and know how to report concerns. ● undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users ● create, re-use, revise and re-purpose digital artefacts for a given audience, with attention to trustworthiness, design and usability ● can understand and apply the fundamental principles and concepts of computer science ● have repeated practical experience of writing computer programs in order to solve problems ● can evaluate and apply information technology

				● are responsible, competent, confident and creative users of information and communication technology ● design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems ● use logical reasoning to compare the utility of alternative algorithms for the same problem ● use two or more programming languages, at least one of which is textual, to solve a variety of computational problems ● can understand and apply the fundamental principles and concepts of computer science ● have repeated practical experience of writing computer programs to solve problems ● are responsible, competent, confident and creative users of information and communication technology. ● design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems ● use logical reasoning to compare the utility of alternative algorithms for the same problem ● use two or more programming languages, at least one of which is textual, to solve a variety of computational problems ● understand a range of ways to use technology safely, respectfully, responsibly and securely, including protecting their online identity and privacy; recognise inappropriate content, contact and conduct and know how to report concerns. ***Summer*** ● can understand and apply the fundamental can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation ● can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems ● understand simple Boolean logic [for example, AND, OR and NOT] and some of its uses in circuits and programming ● understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems

				• understand how instructions are stored and executed within a computer system • Understand that there are different programming languages, of which Small Basic is one. • Be able to write a basic program by breaking a task down into instructions. • Understand what is meant by 'user input' • Know what is meant by 'variable' • Be able to link user input with a variable • Understand how programming languages can use graphics as well as text • Explain how variables can be used • Be able to demonstrate an understanding of computational thinking • Be able to respond effectively to feedback • Be able to use IF and ELSE statements accurately. • Be able to break down a process into instructions which have different outcomes depending on the input. • Understand what ELSEIF is used for. • Understand what is meant by a loop • Know why loops are used to make programs more efficient • Be able to change the number of times a loop runs and explain what it will do to a program • Recognise that a while loop can be used as well as a for loop • Understand the difference between a while loop and a for loop • Be able to explain why a while loop could be used efficiently • Recognise that a while loop can be used as well as a for loop • Understand the difference between a while loop and a for loop • Be able to explain why a while loop could be used efficiently