Singapore Examinations and Assessment Board

CAMBRIDGE
International Education

Singapore–Cambridge Secondary Education Certificate (2027)

G2 Computing
(Syllabus K237)

(First year of examination in 2027)

CONTENTS

AIMS

The aims of the G2 Computing syllabus are to:

1 acquire knowledge and understanding of the concepts of computer systems, networks, application software and programming

2 develop and apply computational thinking skills such as abstraction and decomposition by creating computational artefacts

3 develop and apply media software skills by using application software

4 develop an appreciation of computing as a creative field together with an awareness of cybersecurity, emerging technology and the impact of computing

5 develop attitudes and 21CC needed to do well in computing such including critical, adaptive and inventive thinking, collaboration, communication as well as perseverance in striving for accuracy and thoroughness.

ASSESSMENT OBJECTIVES

The examination will assess candidates':

AO1 knowledge and understanding of computing concepts, application software and the impact of computing

AO2 application of knowledge and understanding of computing concepts, application software and the impact of computing

AO3 practical application of skills in using a range of software to produce computational solutions.

Candidates will demonstrate understanding of computing and networking concepts, application software and the impact of computing. They will use relevant application software to produce computational solutions in the form of spreadsheets and charts, as well as demonstrate computational thinking through analysing and debugging programs. Candidates will also apply their skills to create computer graphics, videos and games.

SCHEME OF ASSESSMENT

All candidates will offer Paper 1 and Paper 2. All questions are compulsory in both papers.

Paper 1 (e-Examination, 1 hour 30 minutes, 70 marks)

This paper will assess candidates' knowledge, understanding and application of concepts and skills in all six modules:

Module 1:	Computer fundamentals
Module 2:	Networking
Module 3:	Impact of computing
Module 4:	Spreadsheets
Module 5:	Media software
Module 6:	Programming

The paper contains two sections. Section A (20 marks) contains 20 multiple-choice questions with 4 choices per question. Section B (50 marks) contains a variable number of short structured questions of variable marks. This paper covers assessment objectives AO1 and AO2.

The paper carries 50% of the total marks and is marked out of 70 marks.

Paper 2 (Lab-based Examination, 2 hours 15 minutes, 80 marks)

This paper, taken with a computer, will assess topics from the following modules:

Module 4:	Spreadsheets
Module 5:	Media software
Module 6:	Programming

Candidates will submit softcopies of the required work for marking. The allotted time includes time for saving the required work in the candidates' computer. This paper covers assessment objective AO3.

This paper carries 50% of the total marks and is marked out of 80 marks.

Summary of details for each paper:

Paper	Mode	Duration	Weighting	Marks	Format	Modules Assessed
1	e-Exam	1h 30m	50%	70	Section A 20 Multiple-choice questions [20 marks] Section B Variable number of short structured questions [50 marks]	All modules
2	Lab-based	2h 15m	50%	80	3 Tasks Media software [~30 marks] Spreadsheets [~25 marks] Programming [~25 marks]	Modules 4–6

SPECIFICATION TABLE

Assessment Objectives		Paper 1	Paper 2	Overall
AO1	Knowledge and understanding	~20%	–	**~20%**
AO2	Application	~30%	–	**~30%**
AO3	Practical application of skills	–	50%	**50%**
TOTAL		**50%**	**50%**	**100%**

USE OF CALCULATOR

An approved calculator may be used in Paper 1 and Paper 2.

CENTRE INFRASTRUCTURE FOR LAB-BASED EXAMINATION

The centre will ensure adequate hardware and software facilities to support the examination of its candidates for Paper 2, which will be administered over at most two shifts on the day of the examination. Each candidate should have the sole use of a laptop for the purpose of the examination. The candidates should be able to access media, spreadsheets and programming software. The centre must ensure that the approved software is installed in all the candidates' laptops.

SYLLABUS CONTENT

The syllabus consists of six modules as follows:

Module 1	Computing fundamentals
Module 2	Networking
Module 3	Impact of computing
Module 4	Spreadsheets
Module 5	Media software
Module 6	Programming

The learning outcomes for each module are shown as follows:

Module 1: Computing fundamentals

1.1: Components

1.1.1 Describe the function of key components of a computer system: its central processing unit (CPU), graphics processing unit (GPU), main memory and secondary storage.
1.1.2 Describe the difference between integrated and dedicated graphics in terms of whether memory is shared between the CPU and GPU.
1.1.3 Describe the use of magnetic and solid-state media for secondary storage in terms of durability, portability, typical capacities, cost and speed.
1.1.4 Compare the sizes of data units: bits, bytes, kilobytes, megabytes, gigabytes, terabytes and petabytes.
1.1.5 Compare specifications of computer systems in terms of their CPU, GPU, main memory and secondary storage.

1.2: Input and output

1.2.1 Identify the input, process and output of a computer application.
1.2.2 Understand that meaningful information is output only after a computer has processed the correct input data.
1.2.3 Give examples of common input devices: keyboards, mice, touchpads, scanners, barcode readers, digital cameras/webcams and microphones.
1.2.4 Give examples of common output devices: display screens/monitors, printers, speakers/headphones and projectors.

Module 2: Networking

2.1: Concepts

2.1.1 Understand that computers in a network can facilitate communication and sharing of documents, hardware and software.
2.1.2 Give examples and state the purposes of common computer network devices: network interface cards, wireless routers and modems.
2.1.3 Understand the difference between wired and wireless communications.
2.1.4 Differentiate between local area networks (LANs) and wide area networks (WANs) based on geographical scope.
2.1.5 Understand the difference between intranets and the internet.
2.1.6 Understand the difference between clients and servers in a client-server network.

2.2: Home networks and the internet

2.2.1 Explain that home networks are examples of LANs and that the internet is an example of a WAN that is formed by connecting many different LANs from around the world together.
2.2.2 Explain that Media Access Control (MAC) addresses are used to direct data within the same LAN while Internet Protocol (IP) addresses are used to direct data across different LANs.
2.2.3 Compare and contrast MAC and IP addresses in terms of purpose and permanence.
2.2.4 Connect a wireless router to a modem correctly such that multiple computers form a LAN that can access the internet.

2.3: Cloud computing

2.3.1 Understand that the cloud refers to computing resources (storage and applications) that are accessed over the internet.
2.3.2 Compare cloud storage to local storage in terms of where files are located and their relative advantages or disadvantages.
2.3.3 Compare cloud-based applications to local applications in terms of relative advantages or disadvantages.

Module 3: Impact of computing

3.1: Technology

3.1.1 Give examples of the impact of computers in:

- Communication: ability to connect people and businesses over long distances
- Education: easy access to online classes and large amounts of information via the internet
- Transportation: widespread access to navigational services via Global Positioning System (GPS) and emergence of self-driving vehicles
- Retail: more reliable tracking of available stock and emergence of self-checkout counters

3.1.2 Describe Artificial Intelligence (AI) as the ability of a computer to perform complex tasks without constant human guidance.
3.1.3 Give examples of tasks that can be performed well by AI: face recognition, voice recognition, image classification, spam filtering, game playing and content generation.

3.2: Responsible use of computers

3.2.1 Understand the online risks associated with scams and malware.
3.2.2 Give examples of malware: viruses, worms, trojans, spyware and ransomware.
3.2.3 Understand how to take measures to prevent falling victim to online risks: use strong passwords, use firewalls, use updated anti-malware programs, identify scam attempts.
3.2.4 Understand how to prevent data loss by making backups for possible recovery in case the originals are damaged.
3.2.5 Understand how to use copyrighted materials responsibly.
3.2.6 Understand the privacy policy and settings of websites before deciding whether to disclose personal information.

Module 4: Spreadsheets

4.1: Cell formats

4.1.1 Set cells to use either a number, currency or percentage format with a specified number of decimal places.
4.1.2 Set cells to use a specified date format.
4.1.3 Use conditional formatting to change the fill and/or font colour of cells based on their contents. [Limited to “greater than”, “less than” and “equal to”]

4.2: Charts

4.2.1 State the purpose of different chart types: bar charts, column charts, pie charts and line charts.
4.2.2 Create bar charts, column charts, pie charts or line charts with data from either a contiguous or non-contiguous range of cells.
4.2.3 Customise chart elements: chart title, data labels, axes, axis titles and legend.
4.2.4 Recognise that modifying a chart's data table will cause a corresponding change to the chart.
4.2.5 State the purpose of combination charts.
4.2.6 Create combination charts. [Limited to combination of line and column charts]

4.3: Formulas

4.3.1 Use mathematical operators (+, -, *, / and ^) in formulas.
4.3.2 Use relational operators (>, >=, <, <=, = and <>) to compare values in formulas.
4.3.3 Use the text concatenation operator (&) in formulas.
4.3.4 Recognise that the value of cells which use formulas will be automatically recalculated when their referenced cells are changed.
4.3.5 Change the view of a spreadsheet to display formulas.
4.3.6 Use absolute and relative cell referencing.

4.4: Functions

4.4.1 Use logical functions to:

- Perform logical OR, AND or NOT [OR, AND, NOT]
- Select between two values based on a logical condition [IF, Up to 1 level of nested IFs]

4.4.2 Use mathematical and statistical functions to:

- Divide, modulo or exponentiate [QUOTIENT, MOD, POWER]
- Round numbers (normal, up, down) [ROUND, CEILING.MATH, FLOOR.MATH]
- Calculate square roots [SQRT]
- Sum numbers (normal, with condition) [SUM, SUMIF]
- Average numbers (normal, with condition) [AVERAGE, AVERAGEIF]
- Calculate median, mode, minimum or maximum of numbers [MEDIAN, MODE.SNGL, MIN, MAX]
- Calculate a value's rank (ascending, descending) [RANK.EQ]
- Calculate n-th largest or smallest value [LARGE, SMALL]
- Count values (numbers only, blank only, non-blank only, with condition) [COUNT, COUNTBLANK, COUNTA, COUNTIF]
- Generate random numbers [RAND, RANDBETWEEN]

4.4.3 Use text functions to:

- Extract characters from text [LEFT, MID, RIGHT]
- Calculate length of text [LEN]
- Concatenate texts [CONCAT]
- Calculate position of text within text (case sensitive, case insensitive) [FIND, SEARCH]

4.4.4 Use look up functions to:

- Look up values from a range of cells using exact matching [HLOOKUP, VLOOKUP]
- Look up values at the intersection of a particular row and column of a range of cells [INDEX]
- Calculate relative position of a value in a range of cells [MATCH]

4.4.5 Use date functions to:

- Determine current date or current date and time [TODAY, NOW]
- Calculate the number of days between two dates [DAYS]

4.5: Sorting and filtering

4.5.1 Sort cells in ascending or descending order based on the contents of a particular column.
4.5.2 Filter data by setting criteria on a column.

4.6: Data validation

4.6.1 State why input data may need to be validated.
4.6.2 Set validation criteria for cells.
4.6.3 Display custom error messages when invalid input data is keyed in.

Module 5: Media Software

5.1: Vector graphics

5.1.1 Explain that vector graphics are created using nodes and paths.
5.1.2 State that vector graphics can be resized without loss of quality.
5.1.3 Create drawings using lines, curves, text, ellipses and polygons.
5.1.4 Move, resize, rotate and flip objects.
5.1.5 Duplicate/copy and delete objects.
5.1.6 Set the front to back arrangement of objects.
5.1.7 Group multiple objects into a single object and ungroup them again.
5.1.8 Recognise solid fills, gradient fills and pattern fills.
5.1.9 Set the fill of objects using a specified colour and style.
5.1.10 Set the transparency of objects such that objects underneath them are visible.
5.1.11 Set the outline of objects using a specified colour and thickness.
5.1.12 Put text to follow a curved path.
5.1.13 Export vector graphics as raster graphics.
5.1.14 Create complex shapes using merging features: union, intersect, fragment, subtract, combine.
5.1.15 Modify objects by manipulating their nodes and node handles directly.

5.2: Raster graphics

5.2.1 State and recognise that raster graphics are composed of individually coloured pixels.
5.2.2 Give PNG, GIF, TIFF, BMP and JPEG as examples of different file formats for raster graphics and state if transparency is supported for each file format.
5.2.3 State that resizing raster graphics can result in a loss of quality.
5.2.4 Explain that the output resolution of raster graphics is measured in dots per inch (dpi) or pixels per inch (ppi) when printed on paper or displayed on a screen respectively.
5.2.5 Adjust the sharpness, brightness, contrast and dimensions (resize and crop) of raster graphics.

5.3: Presentations and videos

5.3.1 Use the slide master feature to achieve a consistent style and layout.
5.3.2 Produce slide presentations based on storyboards.
5.3.3 Understand that frames are individual images in a video.
5.3.4 Create a video from still images and videos with text, transitions and sound.
5.3.5 State that videos with higher frame rates may take up more space but may also appear smoother than videos with lower frame rates.

Module 6: Programming

6.1: Basics

6.1.1 Recognise that visual programming languages use graphical symbols to develop programs.
6.1.2 Interpret flowcharts to understand a program's sequence of events.
6.1.3 State that the purpose of variables is to store values.
6.1.4 Create and name variables.
6.1.5 Initialise and update the values of variables.
6.1.6 Use conditional instructions (if and if-else). [Up to 1 level of nested ifs]
6.1.7 Use basic loops (repeat, forever, for).
6.1.8 Use conditional loops (repeat-until, while).
6.1.9 Use logical (or, and, not) and relational operators (>, < and =) in conditional instructions and/or loops.
6.1.10 Generate and use random numbers in programs.
6.1.11 Use mathematical operators (+, -, * and /) in programs.

6.1.12 Identify and correct errors in programs.

6.1.13 Perform the following list/array operations:

- Create and initialise a list/array
- Insert/delete item at given index of list/array
- Read/write item at given index of list/array
- Check whether an item is in or is not in list/array
- Join two separate lists/arrays

6.1.14 Create and use functions/custom blocks to represent more complicated instructions that are made up of multiple blocks

6.2: Game programming

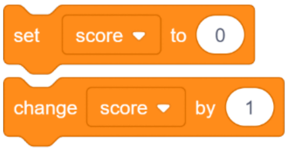
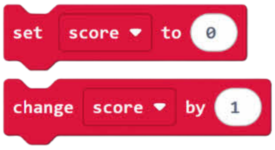
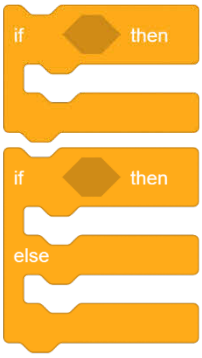
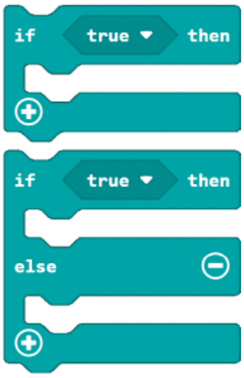
6.2.1 Recognise that points on the stage can be represented using their x and y coordinates.
6.2.2 State that code consists of instructions to be executed by a sprite or the stage.
6.2.3 Recognise that multiple sets of code can be executed at the same time.
6.2.4 Position sprites at a specified location and orientation.
6.2.5 Move and rotate sprites.
6.2.6 Start and stop the execution of code.
6.2.7 Insert wait time between the execution of two instructions.
6.2.8 Insert additional backdrops to the stage by choosing from the library, importing from a file or drawing with the built-in editor.
6.2.9 Switch between the stage's backdrops.
6.2.10 Create and name sprites.
6.2.11 Change the size of sprites.
6.2.12 Insert additional costumes for a sprite by choosing from the library, importing from a file or drawing with the built-in editor.
6.2.13 Switch between a sprite's costumes.
6.2.14 Show and hide sprites.
6.2.15 Display text as either a speech or thought bubble.
6.2.16 Play sounds for an object.
6.2.17 Understand how to record and store digital voice.
6.2.18 Understand how to edit sound clips by performing trim, insert and volume control operations.
6.2.19 Display and hide the values of variables.
6.2.20 Prompt for and accept text input.
6.2.21 Send a message to trigger other objects to start executing their code.
6.2.22 Set key presses and/or mouse clicks to trigger execution of code.
6.2.23 Use contact between sprites and/or coloured areas of objects in conditional instructions and/or loops.

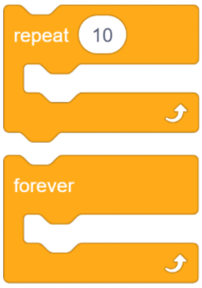
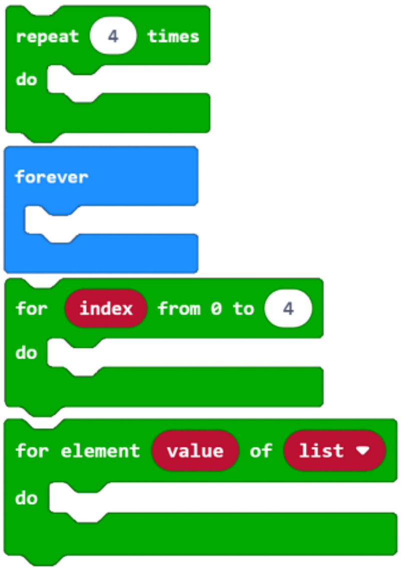
6.3: Microcontrollers*

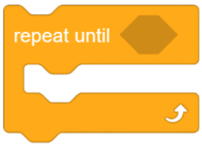
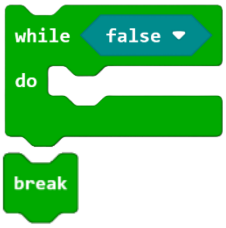
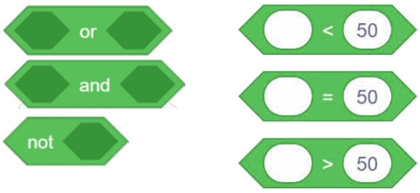
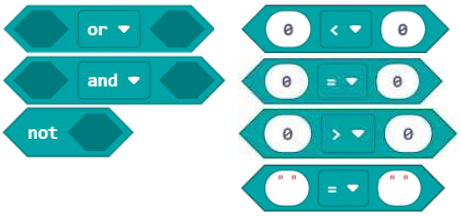
6.3.1 Load a program onto a microcontroller.
6.3.2 Use blocks to read data from built-in input components (light sensor, temperature sensor, timer, accelerometer, compass, buttons, radio) and control built-in output components (LED display, radio).
6.3.3 Connect external input components (buttons, potentiometers, ultrasonic sensors, motion sensors, infrared sensors) and output components (single LEDs, LED strips, motors, buzzers) to the microcontroller.
6.3.4 Use built-in blocks (e.g., digital and analogue input/output pins) or custom blocks to read data from external input components and control external output components.
6.3.5 Use computational thinking (abstraction, algorithmic thinking, decomposition, generalisation, pattern recognition), collaboration, creative thinking and communication skills to identify potential needs and develop prototypes that address those needs with a microcontroller and input/output components.

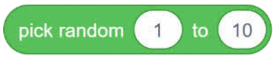
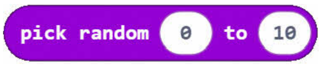
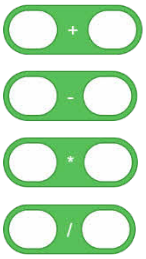
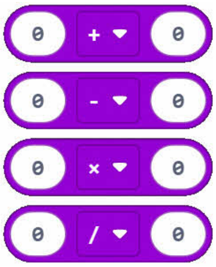
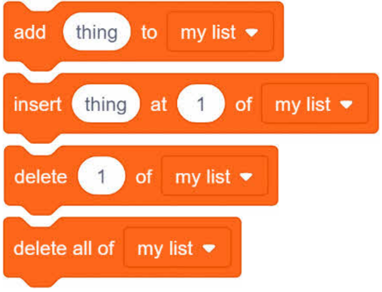
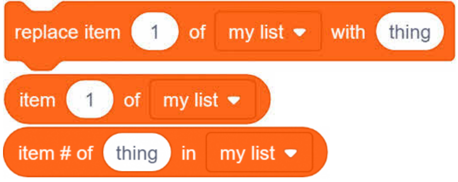
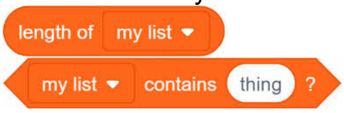
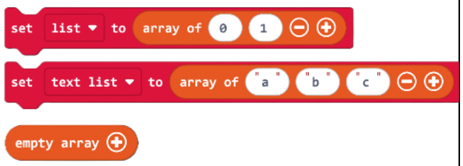
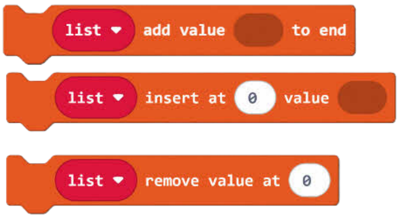
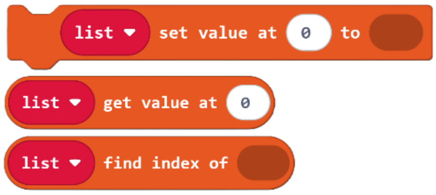
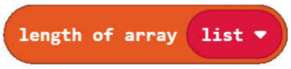
* *To be assessed in G2 Computing Paper 1 only*

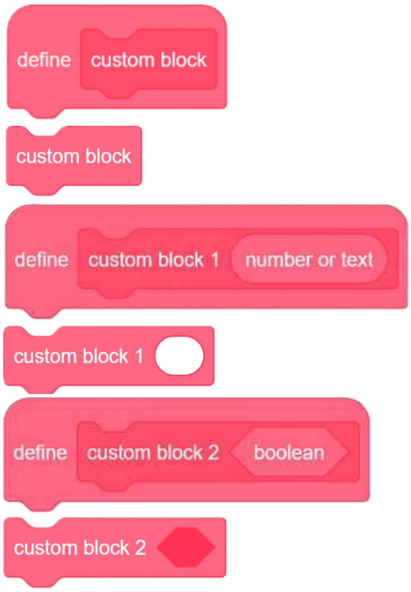
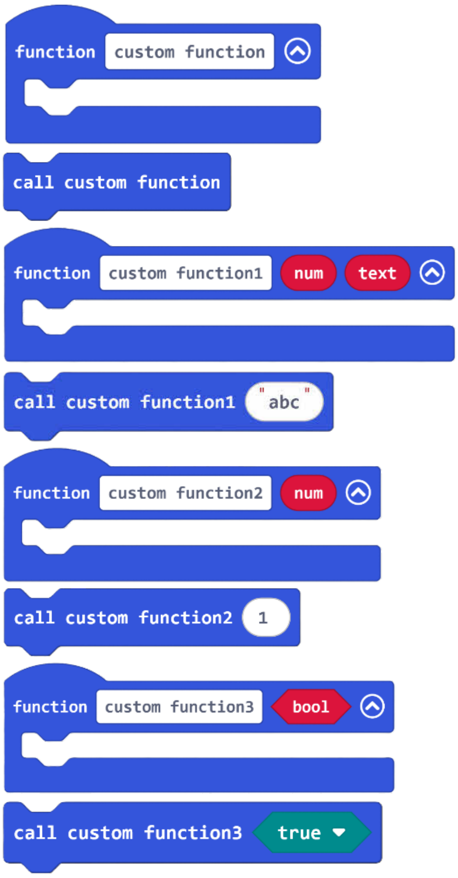
List of Examinable Programming Blocks

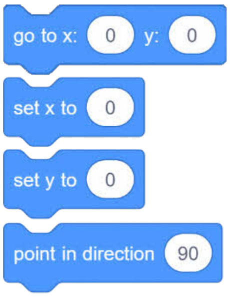
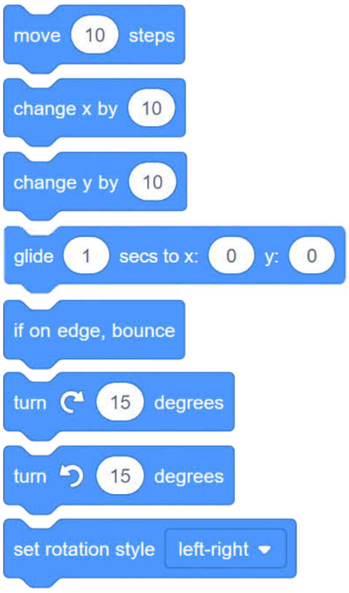
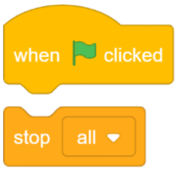
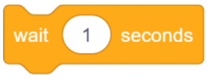
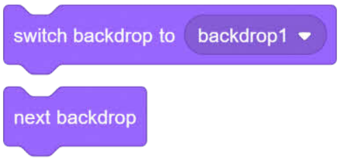
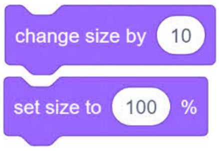
Learning Outcomes 6.1: Basics	**Scratch Blocks (Paper 1 and 2)**	**MakeCode Blocks (Paper 1 only)**
6.1.5 Initialise and update the values of variables.		
6.1.6 Use conditional instructions (if and if-else). [Up to 1 level of nested ifs]		

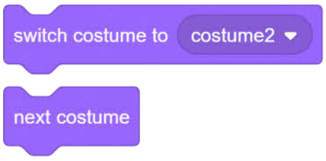
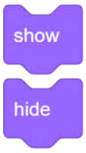
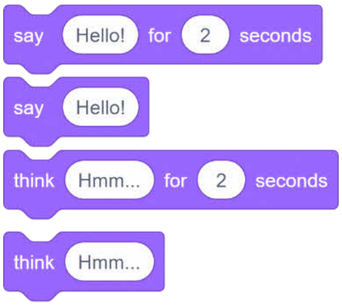
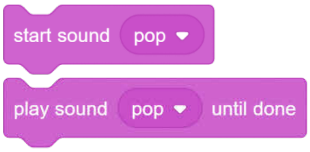
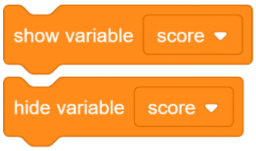
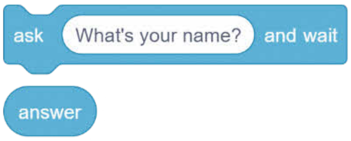
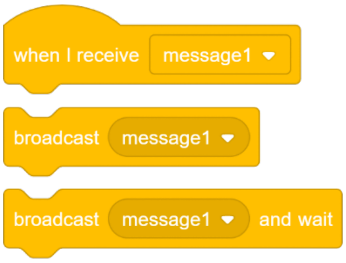
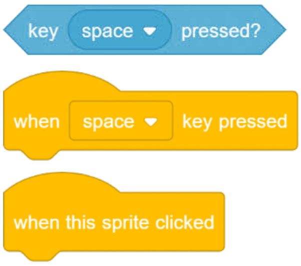
6.1.7 Use basic loops (repeat, forever, for).		 repeat 4 times
		do
		forever
		for index from 0 to 4
		do
		for element value of list
		do

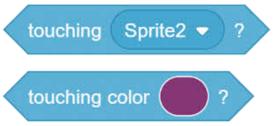
6.1.8 Use conditional loops (repeat-until, while).		
6.1.9 Use logical (or, and, not) and relational operators (>, < and =) in conditional instructions and/or loops.		

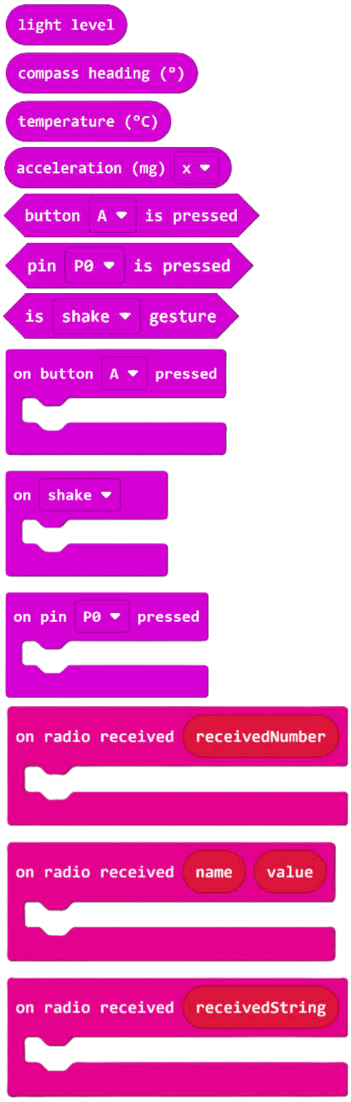
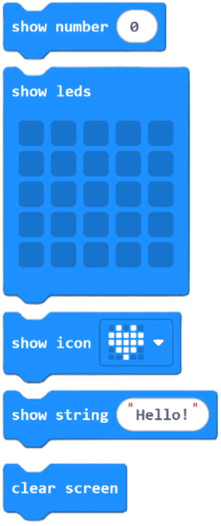
Learning Outcomes 6.1: Basics	**Scratch Blocks (Paper 1 and 2)**	**MakeCode Blocks (Paper 1 only)**
6.1.10 Generate and use random numbers in programs.	 pick random 1 to 10	 pick random 0 to 10
6.1.11 Use mathematical operators (+, -, * and /) in programs.	 + - * /	 0 + 0 0 - 0 0 × 0 0 / 0
6.1.13 Perform the following list/array operations: • Create and initialise a list/array • Insert/delete item at given index of list/array • Read/write item at given index of list/array • Check whether an item is in or is not in list/array • Join two separate lists/arrays	Note: For Scratch, a list is created using the “Make a List” feature. Insert/delete item at given index:  add thing to my list insert thing at 1 of my list delete 1 of my list delete all of my list Read/write item at given index:  replace item 1 of my list with thing item 1 of my list item # of thing in my list Check if list/array contains an item:  length of my list my list contains thing ?	Create and initialise a list/array:  set list to array of 0 1 set text list to array of "a" "b" "c" empty array Insert/delete item at given index:  list add value to end list insert at 0 value list remove value at 0 Read/write item at given index:  list set value at 0 to list get value at 0 list find index of Check if list/array contains an item:  length of array list

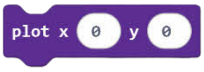
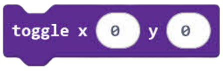
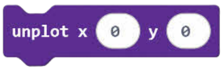
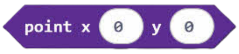
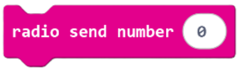
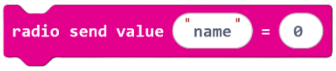
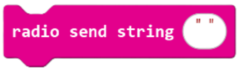
Learning Outcomes 6.1: Basics	**Scratch Blocks (Paper 1 and 2)**	**MakeCode Blocks (Paper 1 only)**
6.1.14 Create and use functions/custom blocks to represent more complicated instructions that are made up of multiple blocks	 define custom block custom block define custom block 1 number or text custom block 1 define custom block 2 boolean custom block 2	 function custom function call custom function function custom function1 num text call custom function1 "abc" function custom function2 num call custom function2 1 function custom function3 bool call custom function3 true

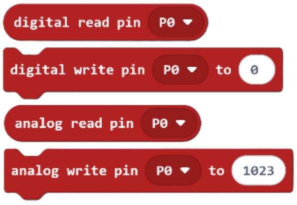
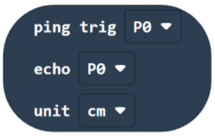
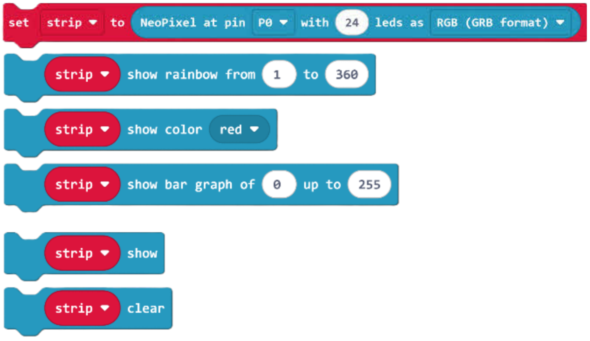
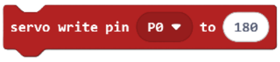
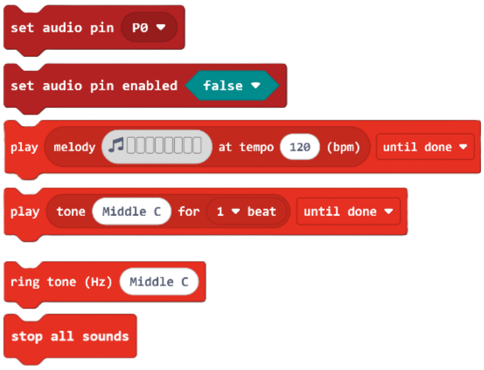
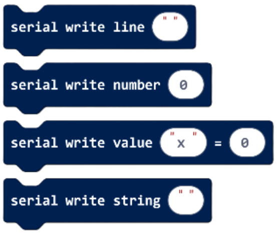
Learning Outcomes 6.2: Game programming	**Scratch Blocks** **(Paper 1 & 2)**
6.2.4 Position sprites at a specified location and orientation.	 go to x: (0) y: (0) set x to (0) set y to (0) point in direction (90)
6.2.5 Move and rotate sprites.	 move (10) steps change x by (10) change y by (10) glide (1) secs to x: (0) y: (0) if on edge, bounce turn ↻ (15) degrees turn ↺ (15) degrees set rotation style [left-right ▾]
6.2.6 Start and stop the execution of code.	 when ⚑ clicked stop [all ▾] (Note: The stop block is not always required to stop execution of scripts)
6.2.7 Insert wait time between the executions of two instructions.	 wait (1) seconds
6.2.9 Switch between the stage's backdrops.	 switch backdrop to [backdrop1 ▾] next backdrop
6.2.11 Change the size of sprites.	 change size by (10) set size to (100) %

Learning Outcomes 6.2: Game programming	**Scratch Blocks (Paper 1 & 2)**
6.2.13 Switch between a sprite's costumes.	 switch costume to costume2 ▾ next costume
6.2.14 Show and hide sprites.	 show hide
6.2.15 Display text as either a speech or thought bubble.	 say Hello! for 2 seconds say Hello! think Hmm... for 2 seconds think Hmm...
6.2.16 Play sounds for an object.	 start sound pop ▾ play sound pop ▾ until done
6.2.19 Display and hide the values of variables.	 show variable score ▾ hide variable score ▾
6.2.20 Prompt for and accept text input.	 ask What's your name? and wait answer
6.2.21 Send a message to trigger other objects to start executing their code.	 when I receive message1 ▾ broadcast message1 ▾ broadcast message1 ▾ and wait
6.2.22 Set key presses and/or mouse clicks to trigger execution of code.	 key space ▾ pressed? when space ▾ key pressed when this sprite clicked

Learning Outcomes 6.2: Game programming	**Scratch Blocks (Paper 1 & 2)**
6.2.23 Use contact between sprites and/or coloured areas of objects in conditional instructions and/or loops.	 touching Sprite2 ▾ ? touching color ● ?

Learning Outcomes 6.3: Microcontrollers	**MakeCode Blocks (Paper 1 only)**
6.3.2 Use blocks to read data from built-in input components (light sensor, temperature sensor, timer, accelerometer, compass, buttons, radio) and control built-in output components (LED display, radio).	Read data from built-in input components:  light level compass heading (°) temperature (°C) acceleration (mg) x ▾ button A ▾ is pressed pin P0 ▾ is pressed is shake ▾ gesture on button A ▾ pressed on shake ▾ on pin P0 ▾ pressed on radio received receivedNumber on radio received name value on radio received receivedString Control built-in output components:  show number 0 show leds show icon ▾ show string "Hello!" clear screen

Learning Outcomes 6.3: Microcontrollers	MakeCode Blocks (Paper 1 only)
	 plot x 0 y 0  toggle x 0 y 0  unplot x 0 y 0  point x 0 y 0  radio send number 0  radio send value "name" = 0  radio send string " "

Learning Outcomes 6.3: Microcontrollers	**MakeCode Blocks (Paper 1 only)**
6.3.3 Connect external input components (buttons, potentiometers, ultrasonic sensors, motion sensors, infrared sensors) and output components (single LEDs, LED strips, motors, buzzers) to the microcontroller. 6.3.4 Use built-in blocks (e.g., digital and analogue input/output pins) or custom blocks to read data from external input components and control external output components.	Buttons, potentiometers, sensors:  digital read pin P0 digital write pin P0 to 0 analog read pin P0 analog write pin P0 to 1023 Ultrasonic sensors:  ping trig P0 echo P0 unit cm LED strips:  set strip to NeoPixel at pin P0 with 24 leds as RGB (GRB format) strip show rainbow from 1 to 360 strip show color red strip show bar graph of 0 up to 255 strip show strip clear Motors/Servos:  servo write pin P0 to 180 Buzzers:  set audio pin P0 set audio pin enabled false play melody at tempo 120 (bpm) until done play tone Middle C for 1 beat until done ring tone (Hz) Middle C stop all sounds Others:  serial write line " " serial write number 0 serial write value "x" = 0 serial write string " "