COMPUTING

SYLLABUS

Secondary

G2

Syllabus K237

Year of Implementation:

Form 2026 with Secondary Three Cohort

Ministry of Education
SINGAPORE

G2 COMPUTING SYLLABUS

For implementation in 2026
First year of examination in 2027

Computer Education Unit
Sciences Branch
Curriculum Planning and Development Division
Ministry of Education
Singapore

CONTENTS

SECTION 1: INTRODUCTION

Value of Computing

Curriculum Framework

Aims of the Syllabus

21st Century Competencies (21CC)

Digital Literacy and Technological Skills (DLTS)

INTRODUCTION

1.1 Value of Computing

The rapid advancement of technology and growing capabilities of Artificial Intelligence (AI) will bring about significant changes to the way we learn, work and play. Companies must embrace digital transformations to survive and strive and individuals must learn new digital skills to remain relevant.

Computing subjects provide upstream support for this effort by providing students with opportunities to acquire useful digital competencies and explore the field of Computing.

Through the G2 Computing subject, our students

- develop basic computational thinking skills along with 21st century competencies,
- develop digital competencies in using digital tools to create digital artefacts, process information, communicate effectively and solve simple problems,
- appreciate the legal, ethical and security issues relating to the use of computers, and
- gain understanding of emerging technologies and the impact of technology on society.

1.2 Curriculum Framework

The design of G2 Computing is guided by the **Computing Curriculum Framework** (shown in the figure below) along with the aims of developing **21st Century Competencies** (see Section 1.4) and **Digital Competencies** (see Section 1.5).

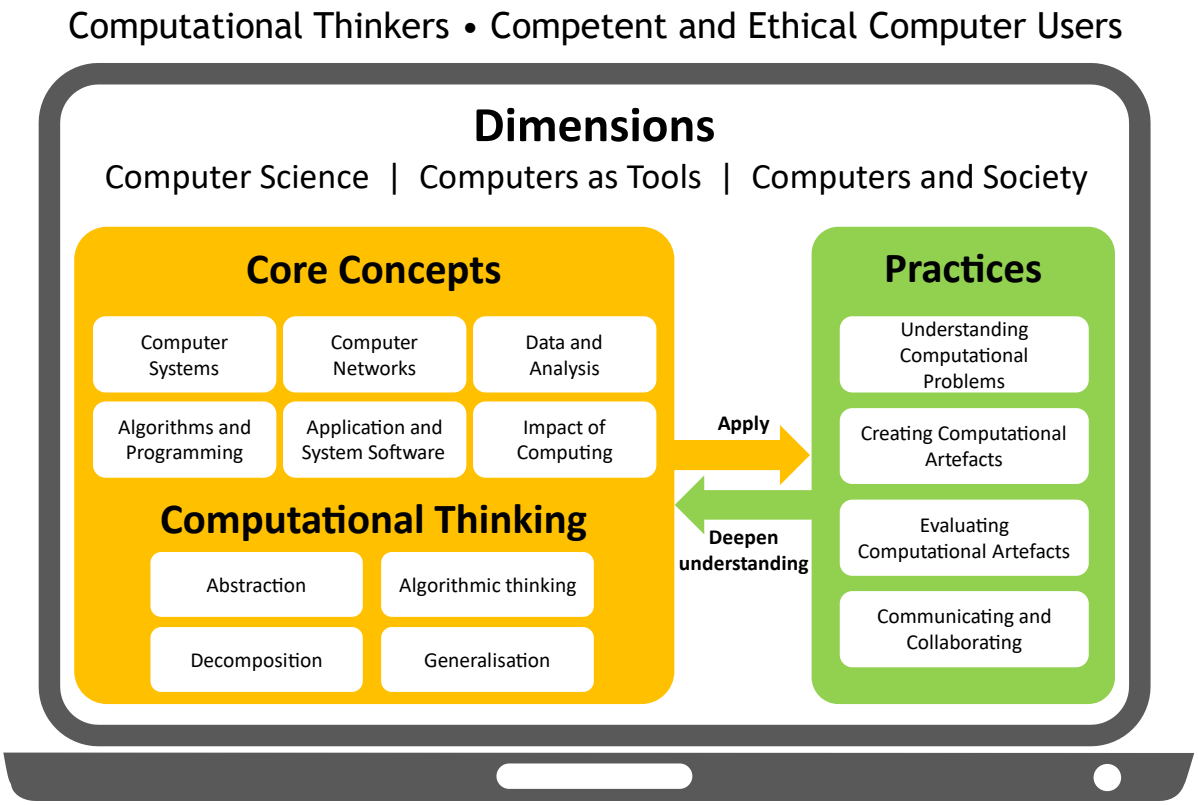

Figure 1: Computing Curriculum Framework (Revised in 2017)

The Computing Curriculum Framework consists of:

- **Vision Statement** for computing education
- **Dimensions** of computing
- **Core Concepts** of computing
- Components of **Computational Thinking (CT)**
- **Practices** of computing practitioners and professionals

An important aspect of the framework is the relationship between the **Core Concepts**, **Computational Thinking** and **Practices**: Core Concepts and Computational Thinking are applied through the Practices, and the Practices will in turn deepen one's understanding of the Core Concepts.

Computational Thinking (CT)[1] has been defined by Jeanette Wing as "the thought process involved in formulating problems and developing approaches to solving them in a manner that can be implemented with an information-processing agent". The practices in the framework aim to support the learning of CT. The components of CT are:

- **Abstraction** - This is the skill of hiding details that are not necessary and leaving only the information that is deemed relevant to developing the solution to the problem.
- **Decomposition** - This is the skill of breaking down a complex problem to smaller and simpler tasks.
- **Generalisation** - This is the skill of identifying patterns and connections and modifying a solution for a specific problem to adapt it to work for a set of similar problems.
- **Algorithmic Thinking** - This involves coming up with the solution to a computational problem and articulating the solution as a sequence of steps or instructions.

The following 2 tables show the alignment of G2 Computing with the **Core Concepts** and **Practices** of the Computing Curriculum Framework respectively.

Table 1: Alignment of G2 Computing topics with the **Core Concepts** in the Framework

Core Concepts	Topics in G2 Computing
Computer Systems	Computing Fundamentals: • components of a computer • inputs and outputs • data units
Computer Networks	Networking: • types of network medium and network organisation • network addresses • network devices • cloud computing
Data and Analysis	Spreadsheets: • organisation and formatting of data • data validation and processing • data analysis using charts

(table continues on the next page)

[1] Wing, J. M. (2006), Computational thinking, Communications of the Association for Computing Machinery,

Core Concepts	Topics in G2 Computing
Algorithms and Programming	Programming: • program development using visual programming • flowcharts and algorithms • problem solving and game making • digital making with microcontrollers
Application and System Software	Media Software: • graphics design • preparation of multimedia presentations • video creation
Impact of Computing	Impact of Computing: • emerging technologies, including examples of AI • cybersecurity risks relating to scams and malware • responsible and ethical use of computers relating to copyright and privacy

Table 2: Alignment of G2 Computing topics with the **Practices** in the Framework

Practices	Tasks in G2 Computing
Understanding Computational Problems (including design tasks)	Students identify and analyse key information about • a design task; or • a computational problem.
Creating Computational Artefacts (including digital artefacts)	Students design and create • digital artefacts such as graphics, presentations and videos; • computational artefacts such as spreadsheets, charts, programs and games; and • physical prototypes using microcontrollers and attached input and output devices.
Evaluating Computational Artefacts (including digital artefacts)	Students test, evaluate and improve • digital artefacts, including exporting and playback of graphics, videos and presentations; • computational artefacts, including debugging and refining of programs and games; and • physical prototypes, including test-runs and hardware troubleshooting.
Communicating and Collaborating	Students work collaboratively in pairs or small groups to create • digital artefacts to complete complex design tasks; • computational artefacts to solve complex computational problems; or • physical prototypes to solve real-world problems.

1.3 Aims of the Syllabus

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

1.4 21st Century Competencies (21CC)

The enhanced Framework for 21st Century Competencies and Student Outcomes ("21CC Framework") in the following figure shows how **Core Values**, **Social-Emotional Competencies**, and **Emerging 21st Century Competencies (E21CC)** support the realisation of **MOE's Desired Outcomes of Education**. G2 Computing supports the development of **E21CCs**, especially **Critical, Adaptive and Inventive Thinking (CAIT)**.

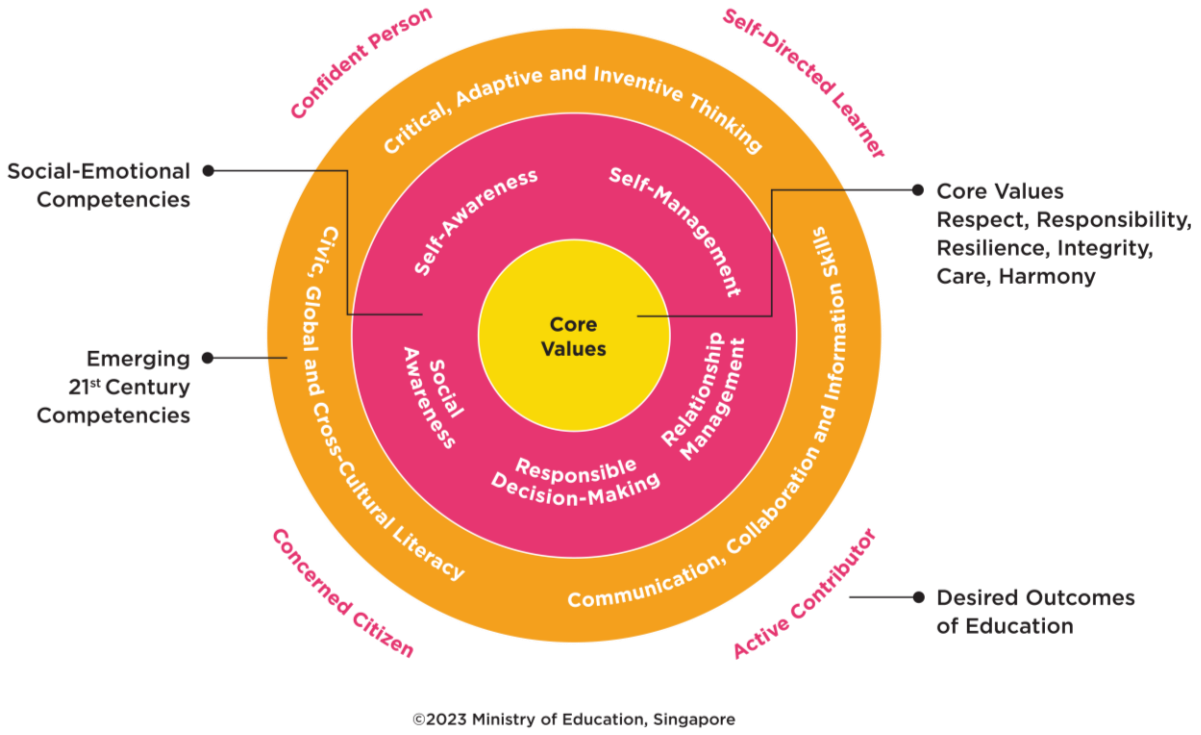

Figure 2: Framework for 21CC and Student Outcomes (Enhanced in 2023)[2]

[2] The enhanced 21CC Framework has been updated with an updated set of learning goals and developmental milestones. Visit go.gov.sg/21cc for more info.

1.5 Digital Literacy and Technological Skills (DLTS)

Digital Literacy (DL) is defined as a set of knowledge, skills and dispositions that would help our learners to be confident, critical and responsible users of digital technologies for information, communication and problem-solving. **Technological skills (TS)** refer to the ability to understand and use specific technologies to solve problems and achieve practical goals.

Under the **EdTech Masterplan 2030**[3], the development of Digital Literacy and Technological Skills (DLTS) in schools is guided by the **Find-Think-Apply-Create (FTAC) frame**, which is in turn unpacked into **9 Digital Competencies (DCs)** as shown in the following figure.

Figure 3: 9 Digital Competencies (DCs) and the Find-Think-Apply-Create (FTAC) frame for Digital Literacy and Technological Skills

The G2 Computing curriculum provides opportunities for the development of Digital Competencies in each of the FTAC components.

[3] For more information on MOE's Edtech Masterplan 2030 and the Find-Think-Apply Create frame for DLTS, visit https://www.moe.gov.sg/education-in-sg/educational-technology-journey/edtech-masterplan

SECTION 2: CONTENT

Overview of Content and Curriculum Time

Learning Outcomes

CONTENT

2.1 Overview of Content and Curriculum Time

The curriculum time for G2 Computing is 3 hours per week over 2 years. A summary of the topics or content explications per module is provided in the following table.

Table 3: Overview of Content and Curriculum Time

Module	Topics
1. Computing Fundamentals	1.1 Components 1.2 Input and Output
2. Networking	2.1 Concepts 2.2 Home Networks and the Internet 2.3 Cloud Computing
3. Impact of Computing	3.1 Technology 3.2 Responsible Use of Computers
4. Spreadsheets	4.1 Cell Formats 4.2 Charts 4.3 Formulas 4.4 Functions 4.5 Sorting and Filtering 4.6 Data validation
5. Media Software	5.1 Vector graphics 5.2 Raster graphics 5.3 Presentations and Videos
6. Programming	6.1 Basics 6.2 Game programming 6.3 Microcontrollers

2.2 Learning Outcomes

Module 1: Computing Fundamentals

Section	Ref.	Learning Outcomes
[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

Section	Ref.	Learning Outcomes
[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

Section	Ref.	Learning Outcomes
[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.
	3.1.4[4]	Show an awareness of emerging technologies such as: · Generative AI · Autonomous Machines · Internet of Things (IoT) · Virtual/Mixed Reality · Quantum Computing
[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	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	Prevent data loss by making backups for possible recovery in case the originals are damaged.
	3.2.5	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

Section	Ref.	Learning Outcomes
[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]

(table continues on the next page)

[4] The content for this learning outcome may be refreshed from time to time and is non-examinable.

Section	Ref.	Learning Outcomes
[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 the 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

Section	Ref.	Learning Outcomes
[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 and contrast 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

Section	Ref.	Learning Outcomes
[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	Record and store digital voice.
	6.2.18	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.

(table continues on the next page)

Section	Ref.	Learning Outcomes
[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.

SECTION 3:

PEDAGOGY

Pedagogical Considerations

Pedagogical Approaches

PEDAGOGY

3.1 Pedagogical Considerations

This section elaborates on the pedagogical considerations used to support the teaching strategies for G2 Computing in alignment to the curriculum framework (see Figure 1 in Section 1.2).

Computational Thinking and Problem-Solving Skills

Computational thinking and problem-solving skills are important for students to take on future challenges in their studies, work and life. Specifically, the nature of Computational Thinking and Problem-Solving lend themselves well to development of Critical, Adaptive and Inventive Thinking (CAIT) in E21CC. To support the aim of developing these skills and the attendant CAIT, the **pedagogical approaches** and **strategies** should provide students with ample opportunities to solve a range of problems of **varying difficulties and contexts** and demonstrate the 4 aspects of computational thinking.

Matching Students' Learning Profile

For learning to be effective, the **teaching pace**, **pedagogical approaches** and **assessment practices** must be **developmentally appropriate** for the profile of students taking the G2 Computing subject. Pedagogical approaches should allow students to engage in designing, creating and evaluating interesting and meaningful computational artefacts such as animations and games. Students **reinforce their learning** when they express knowledge through **creating artefacts** and receive opportunities to **reflectively analyse** their work and the knowledge they have acquired. When students find meaning in learning, they are motivated and challenged, and take ownership of their learning.

Provision of Authentic Contexts

Authentic learning is highly recommended for students taking the G2 Computing subject. Learning activities that **mirror real-world tasks** promote higher levels of engagement as students are required to **actively apply concepts, skills and knowledge** to create computational artefacts (e.g. setting up a spreadsheet to analyse test results) to solve real-world problems. This learning would also be transferable and allows them to apply their learning experiences in new real-world situations which they may face during studies, work and life in the future. The use of authentic contexts will also help to develop Civic and Global Literacy in E21CC.

3.2 Pedagogical Approaches

Based on the above considerations and guided by the **Singapore Curriculum Philosophy (SCP)**, the central pedagogical approaches for the G2 Computing subject are the 'learning through doing' and 'problem-driven' approaches. See the following table for the key features.

Table 4: Key features of 'learning through doing' and 'problem-driven' approaches

Learning Through Doing	Problem-Driven
Students design and create digital and computational artefacts.	Students work on problems which are based on authentic contexts.
Students work collaboratively to design and generate solutions to tasks/problems.	Students understand and identify key information from the description of a computational problem.
Students examine computer programs (i.e. lines of codes) to identify bugs and correct them.	Students solve problems systematically by using decomposition and generalisation.

These two pedagogical approaches are aligned to several **teaching actions/ considerations** that are specific to G2 Computing. In the **Singapore Teaching Practice (STP)**, these teaching actions/ considerations articulate **24 teaching areas** that underpin **4 fundamental teaching processes** as shown in the 4 quadrants of the following figure.

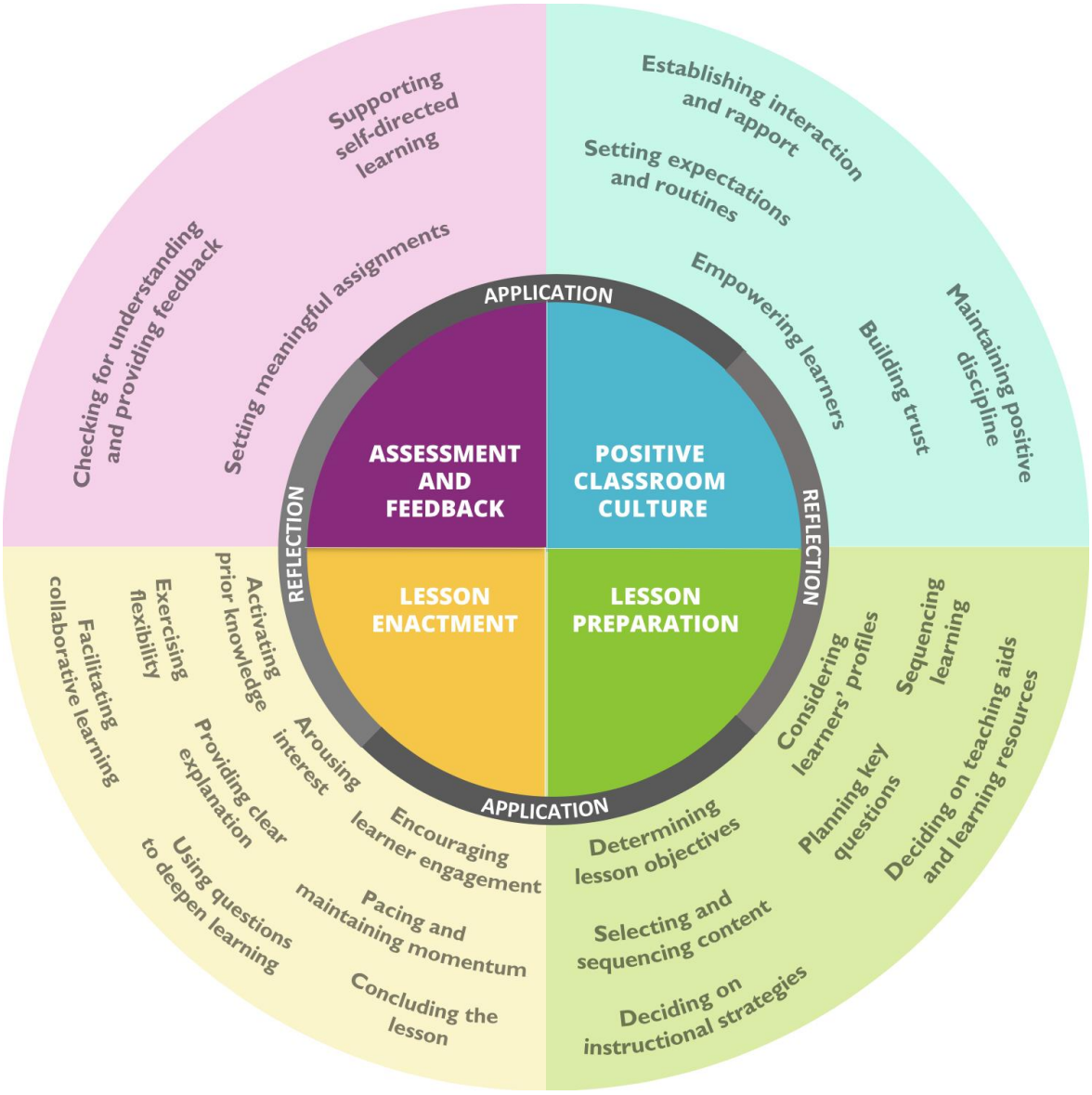

Figure 4: Pedagogical Practices of the Singapore Teaching Practice (STP)

SECTION 4: ASSESSMENT

School-Based Assessment

National Examination

ASSESSMENT

4.1 School-Based Assessment

The assessment for G2 Computing is guided by the **three fundamental beliefs** about assessment in the Singapore Curriculum Philosophy (SCP):

1) Assessment is integral to the teaching and learning process;
2) Assessment begins with clarity of purpose; and
3) Assessment provides feedback to move learning forward and improve teaching practices.

The intent and purpose of **assessment and feedback** are to:

- check for understanding to ascertain the gaps between students' understanding and the desired learning outcomes, and to provide purposeful and meaningful feedback to students;
- design and facilitate self-directed learning activities to reinforce, consolidate and extend learning; and
- set meaningful assignments to inform teaching and support learning.

A **balanced** assessment system consists of both **Assessment for Learning (AfL)** and **Assessment of Learning (AoL)**. Teachers and students may work collaboratively towards more formative- or summative-oriented assessment purposes.

Examples of tasks that are suitable for learning and assessment in G2 Computing:

(i) **Skill-Building Tasks**: Bite-size activities within or outside curriculum time that allow students to practice certain skills, with or without teacher guidance.

(ii) **Problem Sets**: Tasks with real-world context which may integrate skills from more than one topic or software.

(iii) **Course Projects**: Hands-on tasks that require students to demonstrate a range of abilities and skills learnt across different modules. These projects may seek to advance students' conceptual understanding and competency.

School-based summative assessment should consist of timed written and practical components. The written assessment may comprise **multiple-choice** and **short-structured questions**. Practical-based assessment may comprise **hands-on tasks** used to assess students' skills learnt in different modules. The format of the school-based assessment papers may take reference from the national examinations.

Teachers may also utilise online learning environments, such as the **Student Learning Space (SLS)** to assess students' learning and encourage self-directed learning.

4.2 National Examination

Assessment Objectives

The examination will assess candidates'

AO1 Knowledge and understanding of computing concepts, application software and impact of computing

AO2 Application of knowledge and understanding to analyse computing problems and communicate computational solutions

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)

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

Paper 2 (Lab-based Examination)

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

Module 4: Spreadsheets
Module 5: Media Software
Module 6: Programming (excluding section 6.3)

Candidates will submit softcopies of the required work for marking. The allotted time includes time for saving the required work in the candidates' computers.

Details of Each Paper

Paper	Mode	Duration	Weighting/ Marks	Format
1	e-Exam	1 h 30 m	50% (70 marks)	Section A 20 Multiple-Choice Questions [20 marks] Section B Short Structured Questions [50 marks]
2	Lab-based	2 h 15 m	50% (80 Marks)	3 Tasks Media Software [~30 marks] Spreadsheets [~25 marks] Programming [~25 marks]

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%**

SECTION 5:

INFRASTRUCTURE

Hardware and Software Requirements

INFRASTRUCTURE

5.1 Hardware and Software Requirements

Every school offering the subject should be resourced with at least 80 computers provisioned and managed under the **Schools' Standard ICT Operating Environment (SSOE)** programme.

The following table lists the required software to be used for G2 Computing.

Table 5: Software Requirements for G2 Computing

Module	Software Required
2. Networking	Google Drive
4. Spreadsheets	Microsoft Excel
5. Media Software	Microsoft PowerPoint
6. Programming	• Scratch • MakeCode for micro:bit