COMPUTING

SYLLABUS

Secondary

G3

Syllabus 7155

Year of Implementation:
From 2024 with Secondary Three Cohort

Ministry of Education
SINGAPORE

© 2024 Curriculum Planning and Development Division.
This publication is not for sale. Permission is granted to reproduce this publication in its entirety for personal or non-commercial educational use only. All other rights reserved.

Updated: December 2024

G3 COMPUTING SYLLABUS

For implementation in 2024
First year of examination in 2025

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)

1. 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 G3 Computing subject, our students:

- develop computational thinking through problem analysis and evaluation,
- develop effective problem-solving and systems thinking as they think algorithmically, with planning and designing of steps to solve tasks,
- appreciate the legal, ethical and security issues relating to the use of computers,
- gain understanding of emerging technologies and the impact of technology on society,
- develop 21st century competencies, which comprise a set of knowledge, skills, and attitudes that prepare them to address the global challenges of an increasingly dynamic and interconnected world by harnessing technology.

1.2. Curriculum Framework

The design of G3 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).

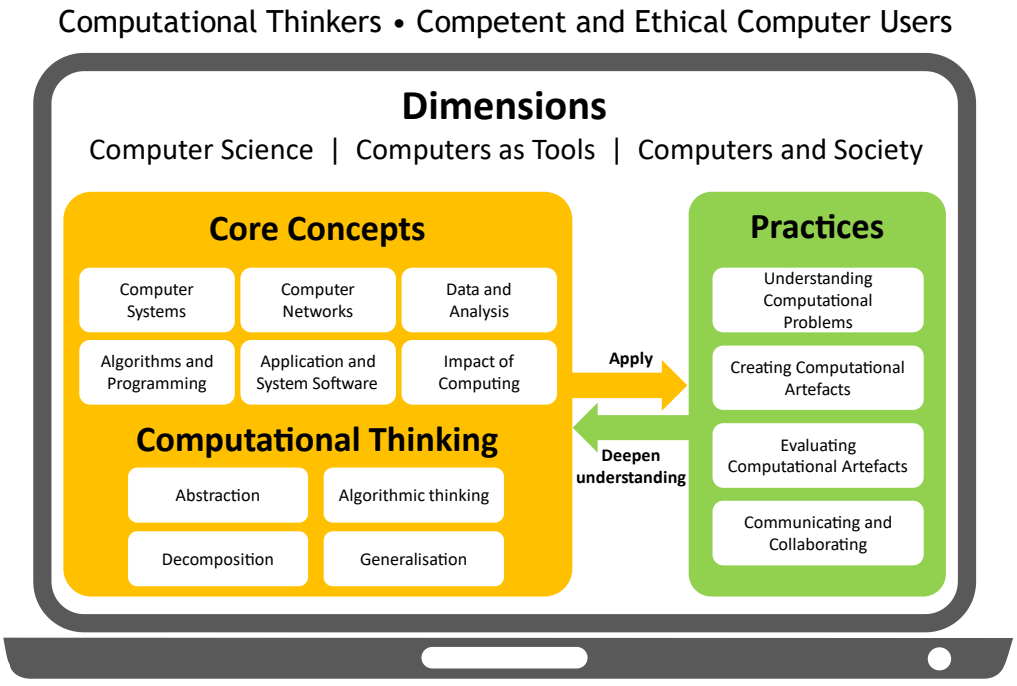

Figure 1: Computing Curriculum Framework (Revised in 2017)

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 G3 Computing with the **Core Concepts** and **Practices** of the Computing Curriculum Framework respectively.

Table 1: Alignment of G3 Computing topics with the **Core Concepts** in the Framework

Core Concepts	**Topics in G3 Computing**
Computer Systems	Computer architecture, data representation and logic gates
Computer Networks	Types of networks, protocols, error detection, home networks and security
Data and Analysis	Data processing and analysis using spreadsheet software
Algorithms and Programming	Python programming, testing, debugging, algorithm design and software engineering

(table continues on the next page)

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

Core Concepts	Topics in G3 Computing
Application and System Software	Application software (spreadsheet and programming software)
Impact of Computing	Intellectual property, online falsehoods and emerging technologies (e.g., AI)

Table 2: Alignment of tasks in G3 Computing with the **Practices** in the Framework

Practices	Tasks in G3 Computing
Understanding Computational Problems	Students understand and identify key information about a complex problem.
Creating Computational Artefacts	Students design and create computational artefacts such as programs and spreadsheets.
Evaluating Computational Artefacts	Students test, evaluate and improve computational artefacts (including debugging and refining programs).
Communicating and Collaborating	Students work collaboratively in pairs or small groups to solve problems and describe / document their solutions.

1.3. Aims of the Syllabus

The G3 Computing syllabus aims to provide students with the foundation to continue with further studies in computing and skills to participate in a rapidly changing technological environment so that the concepts and skills learnt would also be applicable in other fields that require computing. Specifically, the syllabus aims to enable students to:

1) Acquire knowledge and understanding of core areas in computing covering concepts of logic, algorithms, data analysis, data representation and networking;
2) Develop and apply computational thinking skills such as abstraction and decomposition to solve real-world problems by designing, writing, testing and debugging programs using a personal computer;
3) Develop an appreciation of computing as a dynamic and creative field including awareness of recent developments in computer systems;
4) Develop an understanding of the social, ethical, legal and economic implications of computing; and
5) Develop attitudes and 21CC needed to do well in computing such as inventive thinking, perseverance, collaboration, communication as well as 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**. G3 Computing supports the development of **E21CCs**, especially **Critical, Adaptive and Inventive Thinking (CAIT)**.

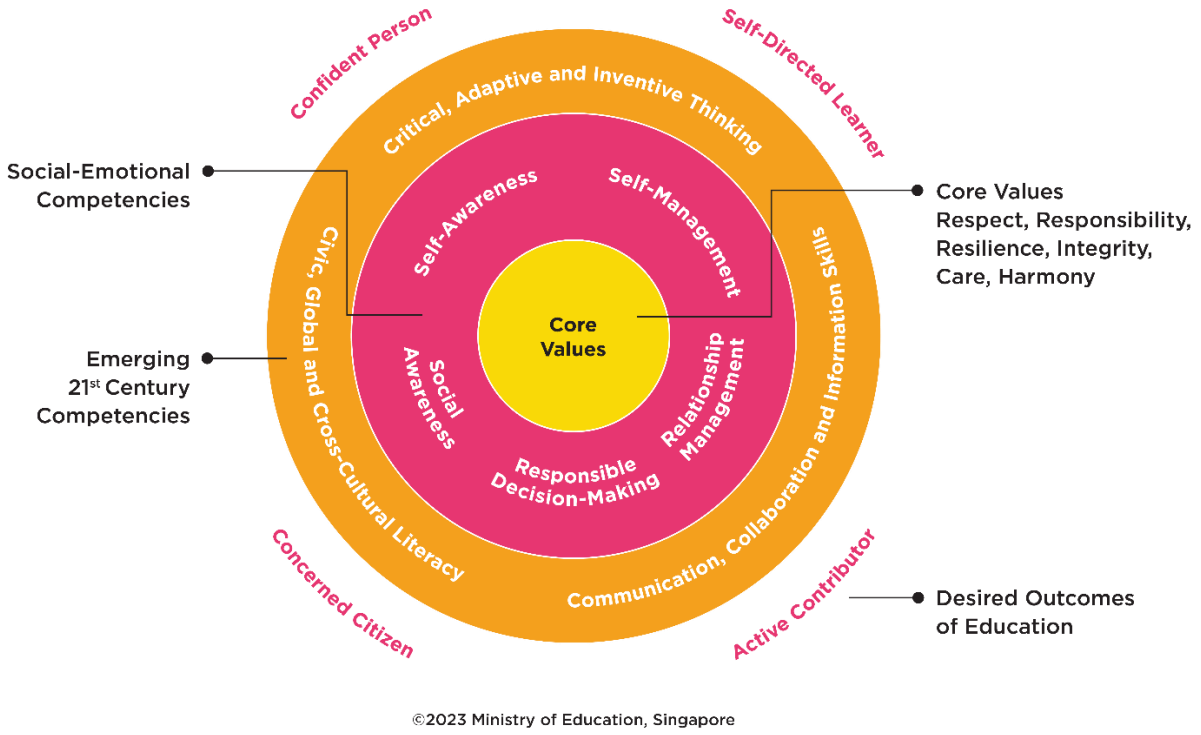

Figure 2: Framework for 21CC and Student Outcomes (Enhanced in 2023)[2]

[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 G3 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

2. CONTENT

2.1. **Overview of Content and Curriculum Time**

The curriculum time for G3 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

Module	**Topics**
1 Computing Fundamentals	1.1 Computer Architecture
	1.2 Data Representation
	1.3 Logic Gates
2 Algorithms and Programming	2.1 Problem Analysis
	2.2 Constructs
	2.3 Python code
	2.4 Testing and Debugging
	2.5 Algorithm Design
	2.6 Software Engineering
3 Spreadsheets	
4 Networking	4.1 Concepts
	4.2 Home Networks and the Internet
	4.3 Security and Privacy
5 Impact of Computing	5.1 General
	5.2 Intellectual Property
	5.3 Communication
	5.4 Emerging Technologies
6 Programming Project	

2.2. Learning Outcomes

Module 1: Computing Fundamentals

1.1: Computer Architecture

Students should be able to:

1.1.1 Perform calculations using bits, bytes, kilobytes, kibibytes, megabytes, mebibytes, gigabytes, gibibytes, terabytes, tebibytes, petabytes and pebibytes.

1.1.2 Describe the function of key components of a computer system: its processor, main memory and secondary storage.

1.1.3 Describe the function of data and address buses in reading from and writing to memory.

1.1.4 Describe different input/output interfaces (USB, HDMI and PCI Express) in terms of typical applications, connectors and speed.

1.1.5 Describe the use of magnetic, optical and solid-state media for secondary storage in terms of durability, portability, typical capacities, cost and speed.

1.2: Data Representation

Students should be able to:

1.2.1 Represent positive whole numbers in binary form.

1.2.2 Convert positive whole numbers from one number system to another - binary, denary and hexadecimal; and describe the technique used.

1.2.3 Use two's complement for a fixed number of bits to represent both positive and negative whole numbers in binary.

1.2.4 Use the example of an 8-bit extended ASCII encoding for English text to explain how information can be represented as bits for storage or processing by a computer.

1.3: Logic Gates

Students should be able to:

1.3.1 Represent logic circuits using either logic circuit diagrams or Boolean statements and convert between the two representations.

1.3.2 Construct the truth table for a given logic circuit (maximum 3 inputs) and vice versa.

1.3.3 Draw symbols and construct truth tables for AND, OR, NOT, NAND, NOR and XOR logic gates.

1.3.4 Manipulate Boolean statements using the associative and distributive properties of certain logical operators and De Morgan's theorem.

1.3.5 Solve system problems using combinations of logic gates (maximum 3 inputs).

Module 2: Algorithms and Programming

2.1: Problem Analysis

Students should be able to:

2.1.1 For a given problem, identify and remove unnecessary details to specify the:

- inputs and the requirements for valid inputs
- outputs and the requirements for correct outputs.

2.2: Constructs

Students should be able to:

2.2.1 Interpret flowcharts to understand the sequence, selection and iteration constructs.

2.3: Python Code

Students should be able to:

2.3.1 Use variables to store and retrieve values.

2.3.2 Use literals to represent values directly in code without using a variable.

2.3.3 Use the built-in functions: input() and print(), to perform interactive input/output using the keyboard and screen.

2.3.4 Use the open() built-in function as well as the read(), readline(), write() and close() methods to perform non-interactive file input/output.

2.3.5 Use the import command to load and make additional variables and functions available for use.

2.3.6 Use Boolean values with the operators: or, and, not.

2.3.7 Use integer and floating-point values with appropriate operators and built-in functions (limited to those mentioned in the Quick Reference Guide) to perform:

- Addition, subtraction, multiplication, division, modulo and exponentiation
- Rounding (normal, up, down, towards zero)
- Calculation of square roots
- Generation of ranges
- Generation of random integers / floats
- Conversion to and from strings

2.3.8 Use string values with appropriate operators, built-in functions and methods (limited to those mentioned in the Quick Reference Guide) to perform:

- Concatenation and repetition
- Extraction of characters and substrings (i.e., indexing and slicing)
- Conversion to upper and/or lower case
- Conversion of single characters to and from ASCII
- Testing of whether characters are letters only, lower-case letters only, upper-case letters only, digits only, spaces only and/or alphanumeric
- Testing of whether the string contains a substring
- Testing of whether the string starts with and/or ends with a substring
- Searching for the location of a substring
- Splitting of string into list of substrings based on either whitespace or a given delimiter
- Calculation of length
- Output formatting

2.3.9 Use list values with appropriate operators, built-in functions and methods (limited to those mentioned in the Quick Reference Guide for Python) to perform:

- Concatenation and repetition
- Extraction of single items and subset of items (i.e., indexing and slicing)
- Testing of whether an item is in the list

- Calculation of length
- Calculation of sum, minimum value and maximum value (provided the list items are all integer or floating-point values)

2.3.10 Use dictionary values with appropriate operators to perform dictionary insertion, query, lookup and deletion.

2.3.11 Use the if, elif and else keywords to implement selection constructs.

2.3.12 Use the for and while keywords to implement iteration constructs.

2.3.13 Write and call user-defined functions that may accept parameters and/or provide a return value.

2.3.14 Distinguish between the purpose and use of local and global variables in a program that makes use of functions.

2.4: Testing and Debugging

Students should be able to:

2.4.1 Produce a trace table by performing a manual dry run and inspecting the value of variables at each step of a program.

2.4.2 Inspect the value of variables at selected steps of a program by inserting print statements.

2.4.3 Locate logic errors by backtracking from a point where unexpected behaviour is observed.

2.4.4 Test programs incrementally as small additions and changes are made during development.

2.4.5 Test small parts of a program by commenting out other parts of the program that are not needed for testing.

2.4.6 Justify the use of data validation and identify the appropriate action to take when invalid data is encountered: asking for input again (for interactive input) or exiting the program (for non-interactive input).

2.4.7 Validate input data for acceptance by performing:
- length check,
- range check,
- presence check,

- format check,
- existence check (i.e., checking for whether input data is already in the system), and/or
- calculation of a check digit

2.4.8 Understand and describe types of program errors: syntax, logic and run-time; and explain why they occur.

2.4.9 Design appropriate test cases to cover normal, error and boundary conditions and specify which type(s) of conditions is/are being tested for each test case.

2.5: Algorithm Design

Students should be able to:

2.5.1 Explain and use the algorithms for:
- Obtaining the minimum or maximum value(s) in a list without using the min() or max() functions
- Calculating the sum or average of values in a list without using the sum() function
- Searching for the location(s) of an item in a list or a character in a string without using the index() or find() methods
- Extracting items from a list or characters from a string based on a given criteria
- Splitting a string into a list based on a given delimiter without using the split() method

2.5.2 Solve problems by breaking them down into smaller and more manageable parts (modular approach).

2.5.3 Solve problems by solving a small version of the problem and gradually extending the solution to bigger versions of the problem (incremental approach).

2.5.4 Use the technique of solving many small instances of a problem manually to identify the generic steps that are needed to solve the problem in general.

2.5.5 Recognise when a new problem is similar to an existing problem that has been encountered before and adapt the corresponding solution to solve the new problem.

2.6: Software Engineering

Students should be able to:

2.6.1 Understand and describe the stages in developing a program: gather requirements, design solutions, write code, test and refine code, deploy code.

2.6.2 Recognise that the sequence of software development stages may not be linear and the use of iterative development may sometimes be more appropriate.

Module 3: Spreadsheets

3.1: Program Features

Students should be able to:

3.1.1 Use appropriate relative, absolute and mixed cell references in formulas so they give the correct results when copied to similar cells in a table.

3.1.2 Use the Goal Seek feature to determine the value needed in a cell for another cell to reach a specified target value.

3.1.3 Use the Conditional Formatting feature to automatically update cell formatting based on one or more rules.

3.2: Functions

Students should be able to:

3.2.1 Use logical functions to perform:

- Logical OR, AND or NOT
- Selection between two values based on a third logical value

3.2.2 Use mathematical and statistical operators and functions to perform:

- Addition, subtraction, multiplication, division, modulo or exponentiation
- Rounding (normal, up, down)
- Calculation of square roots
- Calculation of sums (normal, with condition)
- Calculation of average (normal, with condition)
- Calculation of median, mode, minimum value or maximum value
- Calculation of a value's rank (ascending, descending)
- Calculation of n-th largest or smallest value

- Counting of values (numbers only, blank only, non-blank only, with condition)
- Generation of random numbers

3.2.3 Use text functions to perform:

- Extraction of characters from the left end, middle or right end of text
- Calculation of text length
- Concatenation of texts
- Calculation of the first position of one text within another text (case sensitive, case insensitive)

3.2.4 Use lookup functions to perform:

- Lookup of values from an unsorted vertical or horizontal table using exact matching
- Lookup of values from a sorted vertical or horizontal table using approximate matching
- Classification of values based on range using approximate matching and a secondary table
- Lookup of values at the intersection of a particular row and column of a cell range
- Calculation of the relative position of a value in a cell range

3.2.5 Use date functions to perform:

- Determination of the current date or the current date and time
- Calculation of the number of days between two dates

The examinable spreadsheet operators and functions are as follows.

Operator	Meaning
+	Addition
−	Subtraction or Negation
*	Multiplication
/	Division
%	Percent
^	Exponentiation
=	Equal to
>	More than
>=	More than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
&	Concatenates two values to produce one continuous text value

Category	Function(s)
Date and Time	DAYS, NOW, TODAY
Text	CONCAT, FIND, LEFT, LEN, MID, RIGHT, SEARCH
Logical	AND, IF, NOT, OR
Lookup	HLOOKUP, INDEX, MATCH, VLOOKUP
Mathematical	CEILING.MATH, FLOOR.MATH, MOD, POWER, QUOTIENT, RAND, RANDBETWEEN, ROUND, SQRT, SUM, SUMIF
Statistical	AVERAGE, AVERAGEIF, COUNT, COUNTA, COUNTBLANK, COUNTIF, LARGE, MAX, MEDIAN, MIN, MODE.SNGL, RANK.EQ, SMALL

Module 4: Networking

4.1: Concepts

Students should be able to:

4.1.1 Define computer networks as systems of two or more computers connected by a transmission medium for the exchange of data.

4.1.2 Describe the difference between wired and wireless transmission media and explain the factors that will determine the use of each medium.

4.1.3 Differentiate between Local Area Networks (LANs) and Wide Area Networks (WANs) based on their geographical scope.

4.1.4 Compare and contrast the client-server and peer-to-peer network architectures in terms of purpose, organisation and bandwidth.

4.1.5 Identify and state common applications of star and mesh topologies in a home network.

4.1.6 Define protocols as standards and rules that govern communication over a network.

4.1.7 Explain that LANs typically use protocols where data is transmitted as individual packets.

4.1.8 Explain the use of parity, checksums, echo checks and automatic repeat requests for detecting errors in packet transmission.

4.2: Home Networks and the Internet

Students should be able to:

4.2.1 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.

4.2.2 Explain that modems are used to provide internet access by converting from the protocols used by an Internet Service Provider (ISP) to the protocols used by LANs.

4.2.3 Explain that computers use network interface controllers to communicate via different transmission media.

4.2.4 Explain that Media Access Control (MAC) addresses are used by network interface controllers, network switches and wireless access points to direct data within the same LAN while Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6) addresses are used by routers to direct data across different LANs.

4.2.5 Compare and contrast MAC, IPv4 and IPv6 addresses in terms of purpose, bit length, degree of permanence and typical representation formats.

4.2.6 Connect a router, a switch and a wireless access point to a modem correctly such that multiple computers form a LAN that can access the internet via wired and wireless transmission media.

4.3: Security and Privacy

Students should be able to:

4.3.1 Compare and contrast security and privacy in terms of what kind of data is being protected, what the data is being protected from and how that protection is enforced.

4.3.2 Explain how human actions threaten security and privacy by causing data corruption (through physical or non-physical means) or exposure of private data.

4.3.3 Explain how anti-malware programs enforce security and privacy by preventing malware from running and removing malware that may be present on a computer.

4.3.4 Explain how firewalls enforce security and privacy by using either hardware or software to monitor packets and decide which packets should be permitted or blocked based on a set of configurable rules.

4.3.5 Explain how encryption enforces security and privacy by making encrypted data appear meaningless without the corresponding secret key.

4.3.6 Explain how the Personal Data Protection Act (PDPA) enforces privacy by legally requiring that organisations do the following when collecting personal data:

- seek consent from the individual;
- disclose the purpose for collecting data when seeking consent; and
- retain the data for only as long as necessary to fulfil the stated purpose

4.3.7 Explain how adware threatens security and privacy by installing itself without the user's knowledge and displaying unwanted advertisements.

4.3.8 Explain how spyware threatens security and privacy by secretly collecting personal information and transmitting this information to attackers without the user's knowledge.

4.3.9 Explain how cookies are typically not malicious but can threaten privacy by tracking a user's browsing history across multiple web sites.

4.3.10 Explain how phishing threatens security and privacy by using emails and fake websites that appear to be from reputable companies to steal personal information.

4.3.11 Explain how pharming threatens security and privacy by intercepting requests to legitimate websites and redirecting them to fake websites while still appearing to use the same address as the legitimate website.

4.3.12 Describe good computing practices that can mitigate the threats posed by adware, spyware, cookies, phishing, pharming and human actions.

4.3.13 Analyse the effects of anti-malware programs, firewalls, encryption and the PDPA against the threats posed by adware, spyware, cookies, phishing, pharming and human actions.

Module 5: Impact of Computing

5.1: General

Students should be able to:

5.1.1 Give examples of the impact of computers in the following industries:

- 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

5.2: Intellectual Property

Students should be able to:

5.2.1 Define intellectual property as creations of the mind that have value but can exist purely as data.

5.2.2 Describe copyright as the legal right of owners to control the use and distribution of their intellectual property under the Copyright Act.

5.2.3 Explain that copyright owners can grant a license to authorise or forbid the use and distribution of their intellectual property under certain conditions.

5.2.4 Identify computer programs as an example of intellectual property and distinguish between proprietary software, freeware, shareware as well as free and open-source software (FOSS) based on their licenses.

5.2.5 Recognise software piracy as the illegal use and distribution of copyrighted computer programs in a manner that is forbidden by their license.

5.3: Communication

Students should be able to:

5.3.1 Explain why the promotion of social media posts based on engagement rate helps to deliver relevant content to users but can also lead to the proliferation of falsehoods.

5.3.2 Explain how the Protection from Online Falsehoods and Manipulation Act (POFMA) enables the government to tackle the spread of fake news by:

- establishing fines and/or prison terms for engaging in prohibited activities;
- optionally requiring offenders to put up a correction notice or to take down the falsehood; and
- identifying sites that repeatedly spread falsehoods.

5.4: Emerging Technologies

Students should be able to:

5.4.1 Describe Artificial Intelligence (AI) as the ability of a computer to perform complex tasks without constant human guidance and improve its performance as more data is collected.

5.4.2 Give examples of common personal and business tasks that can be performed well by AI: face recognition, voice recognition, image classification and spam filtering.

5.4.3 Define Machine Learning (ML) as a technique used in AI and explain the difference between ML and traditional programming.

5.4.4 Use the nearest neighbour method for a classification task with two quantitative features to demonstrate the basic principles behind ML.

5.4.5 Explain how unethical use of AI or using AI with biased data can lead to negative consequences.

5.4.6 Show an awareness of emerging technologies.[4]

[4] This is a non-examinable Learning Outcome.

SECTION 3: PEDAGOGY

Pedagogical Considerations
Pedagogical Approaches
Performance Tasks

3. PEDAGOGY

3.1. Pedagogical Considerations

This section elaborates on the pedagogical considerations used to support the teaching strategies for G3 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 G3 Computing subject. Pedagogical approaches should allow students to engage in designing, creating and evaluating interesting and meaningful computational artefacts such as programs and spreadsheets. 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 G3 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 G3 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 G3 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.

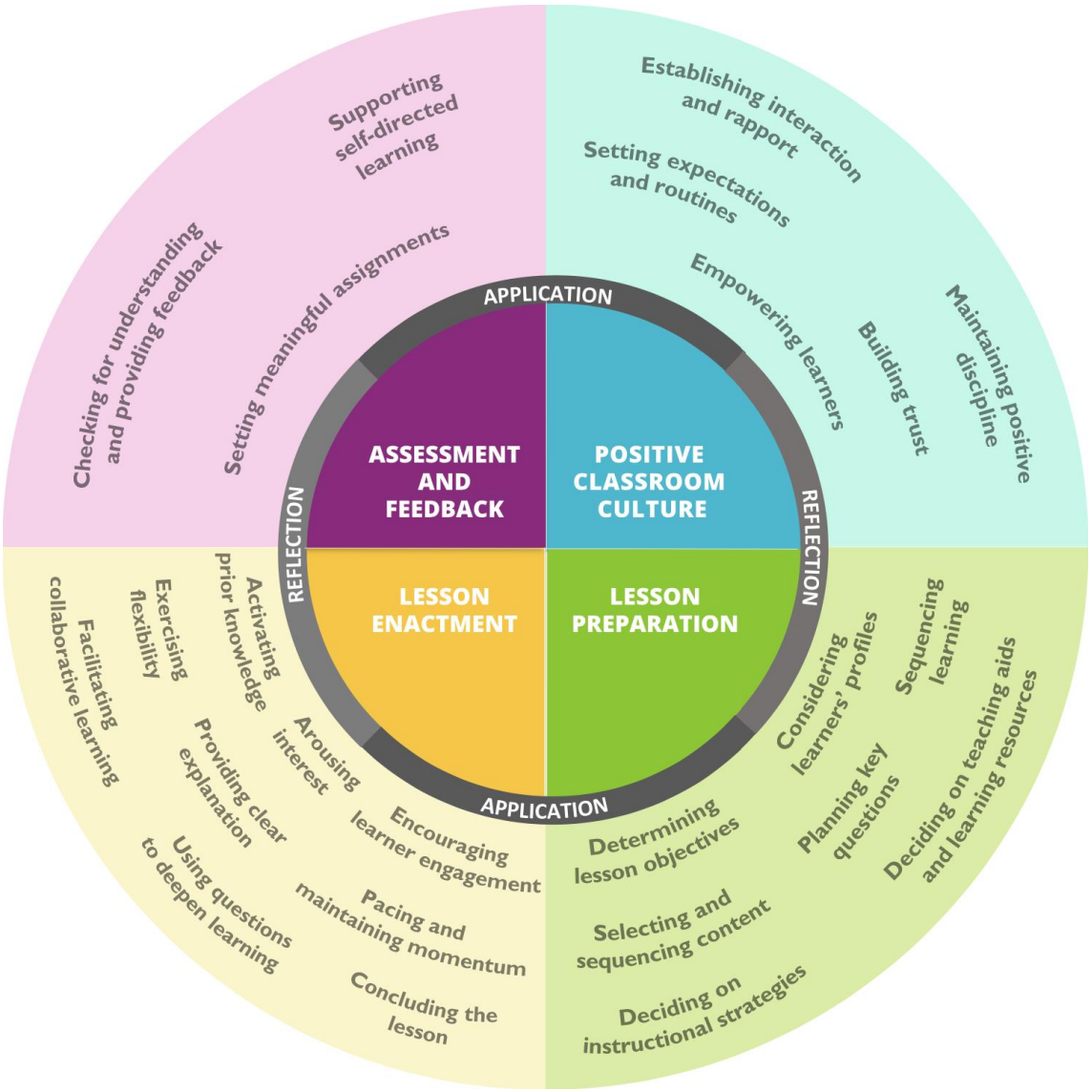

Figure 4: Pedagogical Practices of the Singapore Teaching Practice (STP)

3.3. Performance Tasks

To provide meaningful assignments to reinforce and extend students' learning, teachers could use the following:

(a) Quick Checks are provided at the end of each section of the textbook to reinforce concepts through short-answer questions. Teachers could use these to quickly check for students' understanding of that section.

(b) Worksheets are used to complement teaching. These could be used as classroom activities where teachers can do together with their students.

(c) Review Questions are provided at the end of the chapter of the textbook to reinforce and consolidate the concepts learnt in the chapter through structured questions. Teachers could assign these after the teaching of a chapter has been completed.

(d) Student Learning Space leverage on the affordances of ICT to enhance students' learning through simulations, games and videos.

(e) Each Mini Task consists of a few guided programming tasks with a real-world context. Teachers could assign these tasks after the teaching of Algorithms and Programming has been completed.

(f) Mini Projects are open-ended tasks carried out at the end of Secondary 3 over 6 weeks. The project will be based on authentic tasks founded on real-world problems. There are 2 proposed implementations for the Mini Project – hardware-based or software-based. In the hardware project, students program microcontrollers to perform real-world tasks such as monitoring temperature within an area. In the software project, students may program a game to move a sprite around to collect tokens. See Table 5.

Table 5: Possible Mini Projects

Software-based	
Pygame module	Creation of simple games
Turtle module	Creation of graphics and simple games
Numpy and Matplotlib modules	Analysis and visualisation of data
Hardware-based	
Raspberry Pi	Creation of simple maker projects (e.g., alarm clock)
	Creation of simple games using the Pygame module and Sense HAT (e.g., controlling game using rotation sensor of Sense HAT)
Micro:bit	Creation of simple maker projects (e.g., counter, dice, wireless messaging, pendulum oscillation counter)
	Creation of simple games (e.g., catch game)

SECTION 4: ASSESSMENT

School-based Assessment
National Examination

4. ASSESSMENT

4.1. School-Based Assessment

The assessment for G3 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 G3 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 core computing concepts, algorithms, techniques, tools and related ethics

AO2 Application of knowledge and understanding to analyse and solve computing problems

AO3 Design, development, testing and refinement of computing solutions

Students will handle and process data in computer systems and demonstrate their understanding on ethical issues when dealing with data. They will demonstrate problem-solving techniques through analysing and writing programming solutions for a range of computing problems in a variety of contexts. Students will also demonstrate computational thinking through the design and development of computing solutions.

Scheme of Assessment

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

Paper 1 (Written examination, 2 hours)
This paper will assess candidates' knowledge, understanding and application of concepts and skills in all the six modules. The questions consist of a mixture of:

- Multiple choice questions (single- and multiple-answer)
- Short-answer questions
- Matching questions
- Cloze passages
- Structured questions

Relevant formulae will be provided for candidates.

The paper carries 60% of the total marks and is marked out of 80 marks.

Paper 2 (Lab-based Examination, 2 hours 30 minutes)
This paper, taken with the use of a computer with access to a spreadsheet, Python and JupyterLab software, will assess topics from the following modules:

- Spreadsheets
- Algorithms and Programming

A quick reference guide for Python will be provided for candidates.

Candidate 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 carries 40% of the total marks and is marked out of 70 marks.

Summary of details for each paper:

Paper	Mode	Duration	Weighting	Marks	Format	Modules Assessed
1	Written	2 h	60%	80	A mixture of • Multiple choice questions (single- and multiple-answer) • Short-answer questions • Matching questions • Cloze passages • Structured questions	All the five modules
2	Lab-based	2 h 30 m	40%	70	• One question on Spreadsheets • Four to five questions on Programming	Module 2: Algorithms and Programming Module 3: Spreadsheets

Specification Table

Assessment Objectives		Paper 1	Paper 2	Overall
AO1	Knowledge and Understanding	~30%	0%	**30%**
AO2	Application	~20%	~20%	**40%**
AO3	Development, testing and refinement	~10%	~20%	**30%**
TOTAL		**60%**	**40%**	**100%**

Use of Calculator

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

SECTION 5: INFRASTRUCTURE

Hardware and Software Requirements

5. 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 G3 Computing.

Table 6: Software requirements for G3 Computing

Module	**Software**
Algorithms and Programming	• JupyterLab Desktop • Mu (for T&L only)
Spreadsheets	• Microsoft Excel
Networking	• Filius (for T&L only)