COMPUTING

SYLLABUS

Pre-University

Higher 2

Syllabus 9569

Implementation starting with
2019 Pre-University One Cohort

Ministry of Education
SINGAPORE

© 2018 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.

CONTENTS

SECTION 1: INTRODUCTION

Desired Outcomes of Education and Learning of Computing
Value of Learning Computing
Design Intent of Syllabus
Curriculum Framework
Aims of Syllabus
21st Century Competencies (21CC) in H2 Computing

1. INTRODUCTION

1.1 Desired Outcomes of Education and Learning of Computing

The Desired Outcomes of Education (DOE) are attributes that educators aspire for every Singaporean to have by the completion of his formal education. These outcomes establish a common purpose for educators, drive our policies and programmes, and allow us to determine how well our education system is doing.

The person who is schooled in the Singapore Education system embodies the DOE. He has a good sense of self-awareness, a sound moral compass, and the necessary skills and knowledge to take on challenges of the future. He is responsible to his family, community and nation. He appreciates the beauty of the world around him, possesses a healthy mind and body, and has a zest for life. In sum, he is

- a confident person who has a strong sense of right and wrong, is adaptable and resilient, knows himself, is discerning in judgment, thinks independently and critically, and communicates effectively;

- a self-directed learner who takes responsibility for his own learning, who questions, reflects and perseveres in the pursuit of learning;

- an active contributor who is able to work effectively in teams, exercises initiative, takes calculated risks, is innovative and strives for excellence; and

- a concerned citizen who is rooted to Singapore, has a strong civic consciousness, is informed, and takes an active role in bettering the lives of others around him.

The learning of H2 Computing is aligned with the DOE. Through applying their knowledge of relevant computing concepts and computational thinking skills, students are able to create solutions to authentic problems. For example, during the *problem definition* phase, students establish clearly what the problem is by determining the scope of the requirements and data flows. During the *problem analysis* phase, students think logically about how the problem can be decomposed into smaller and more manageable parts. During the *design* phase, students apply abstraction to focus on important parts of the problem while hiding unnecessary details as they think about possible solutions. During the *development of the solution* phase, they actualise the design by creating an algorithm that solves the problem. The last phase of *computer-based solution* requires students to translate an algorithm into a computer-based program using a programming language. Finally, they also need to test the program to ensure that it works as designed.

These authentic learning experiences encourage students to become critical thinkers and innovators in designing solutions to complex problems. They are also able to develop perseverance and resilience through rigorous debugging and refinement of their own programs. Besides developing these qualities, there are also opportunities for students to

think critically, evaluate information sources, collaborate with others and communicate effectively. The A-level Computing syllabus thus offers varied and enriched learning opportunities centred around the DOE by building useful content knowledge and developing the necessary skills and attitudes related to computing in students.

1.2 Value of Learning Computing

The discipline of computing[1] is defined as the systematic study of algorithmic processes that describe and transform information: their theory, analysis, design, efficiency, implementation, and application. The fundamental question underlying all computing is "What can be (efficiently) automated?"

The study of H2 Computing develops in students 21st century competencies, a set of knowledge, skills and attitudes to prepare them to address the global challenges of an increasing dynamic and interconnected world by harnessing technology.

Technology is so pervasive and intertwined into our daily lives to the extent that almost every aspect of the modern world is linked or dependent on it. Through the study of Computing, students will gain insights as to how common computing devices encountered in our daily lives actually function, and to develop an appreciation for computing innovation. They will be better able to understand the efforts underpinning technological advancements, as well as to discuss the benefits and negative consequences brought about by computing.

A Computing student will develop computational thinking and systems thinking during the course of study essential to problem solving. A Computing student will be able to effectively integrate the use of both hardware and software to create new artefacts to solve existing problems, as well as identify processes and tasks that can be automated by a computer, thereby increasing the efficiency of current processes.

<u>Computational Thinking and Problem Solving skills</u>

Core to the subject of Computing is computational thinking. Computational thinking is the process of problem solving which involves formal reasoning, logic and algorithmic thinking. Students will develop computational thinking through the processes of problem analysis, and the design and development of solutions for a variety of given problem situations. In a problem situation, students analyse and identify the tasks that required computing solutions. Students will consider possible empirical models, analyse data and evaluate information that will help them in the problem solving process. They think algorithmically as they plan and design the set of steps to solve each task, and engage in logical and analytical thinking as they develop, test and evaluate the solutions.

These skills also support the development of 21CC in the domain of Critical and Inventive Thinking (CIT); and Communication, Collaboration and Information Skills (CCI) which is

[1] D.E., Corner, & David, Gries, & Micheal, Mulder, & Allen Tucker, & A.Joe, Turner, & Paul R., Young. *Computing as a Discipline.* New York, USA: Communications of the ACM, Volume 32 Issue 1, Jan. 1989, Pages 9-23

elaborated in **Section 1.6 Table A1**.

1.3 Design Intent of Syllabus

The design of the H2 Computing curriculum took into consideration the key findings from the environment scans of local and international syllabi, and the value proposition of Computing for pre-university school curriculum. The core content area is organized so as to allow students to apply knowledge, design and develop computing solutions.

The following key ideas are presented in the A-Level H2 Computing curriculum to integrate Computing concepts, skills and processes as a coherent whole:

- Computing is a study of problems that can be effectively automated.
- Computational thinking is the process of identifying computational problems and developing solutions to them.
- Computational thinking involves problem definition, problem analysis, design and development of solution and creating a computer-based solution.
- Algorithms are tools for developing and expressing solutions to computational problems.
- Programming is a creative process that produces computational artefacts.
- Computing devices and the networks that interconnect them enable and foster computational approaches to solving problems.

1.4 Curriculum Framework

The design of the H2 Computing curriculum is guided by the Computer Education Framework. The aim is to provide a balanced coverage between theory and practice through the learning of fundamental Computing concepts and principles, as well as the application of logical reasoning and problem-solving skills to practical contexts. The framework consists of three dimensions: Computer as a Tool, Computer in Society and Computer as a Science. These three dimensions undergird the broad ideas of systems thinking and computational thinking inherent in the study of Computing.

The computer education framework comprises three dimensions (Figure 1):

- Computer as a Science
- Computer as a Tool
- Computer in Society

Computer as a Science

The dimension of Computer as a Science looks into the scientific aspect of computer science, focusing on the core components of computational and systems thinking.

Computational thinking develops students' skills in problem solving through algorithmic thinking and design. Acquisition of programming language skills is usually a part of this area of learning. Computational thinking, as defined by Jeannette M. Wing[2], is a way people solve problems and that it is not about trying to get people to think like computers[3]. This often involves thinking and problem-solving processes to reformulate a seemingly difficult task into one we know how to solve. Thus, computational thinking, in her opinion, is a fundamental skill for everyone, not just for computer scientists.

Systems thinking develops students in the design and creation of systems and solutions through processes in problem definition, system analysis, and systems design.

Computer as a Tool

The dimension of Computer as a Tool looks mainly at the utilitarian aspect of computing and ICT. At the heart of it are the use of the computer and the use of computer applications. Use of computer exposes students to the hardware, the technology and related devices and peripherals that open up ways for work, play and living.

Use of computer applications focuses on the mastery of productivity, communications and creative tools to complete tasks for specific purposes. Common examples include word processing, spreadsheets, graphics, emails, animation and web design.

[2] Jeannette M. Wing is the President's Professor of Computer Science and head of the Computer Science Department at Carnegie Mellon University.

[3] J. M. Wing. Computational Thinking. Communications of the Association for Computing Machinery (ACM), March 2006, Vol. 49(3).

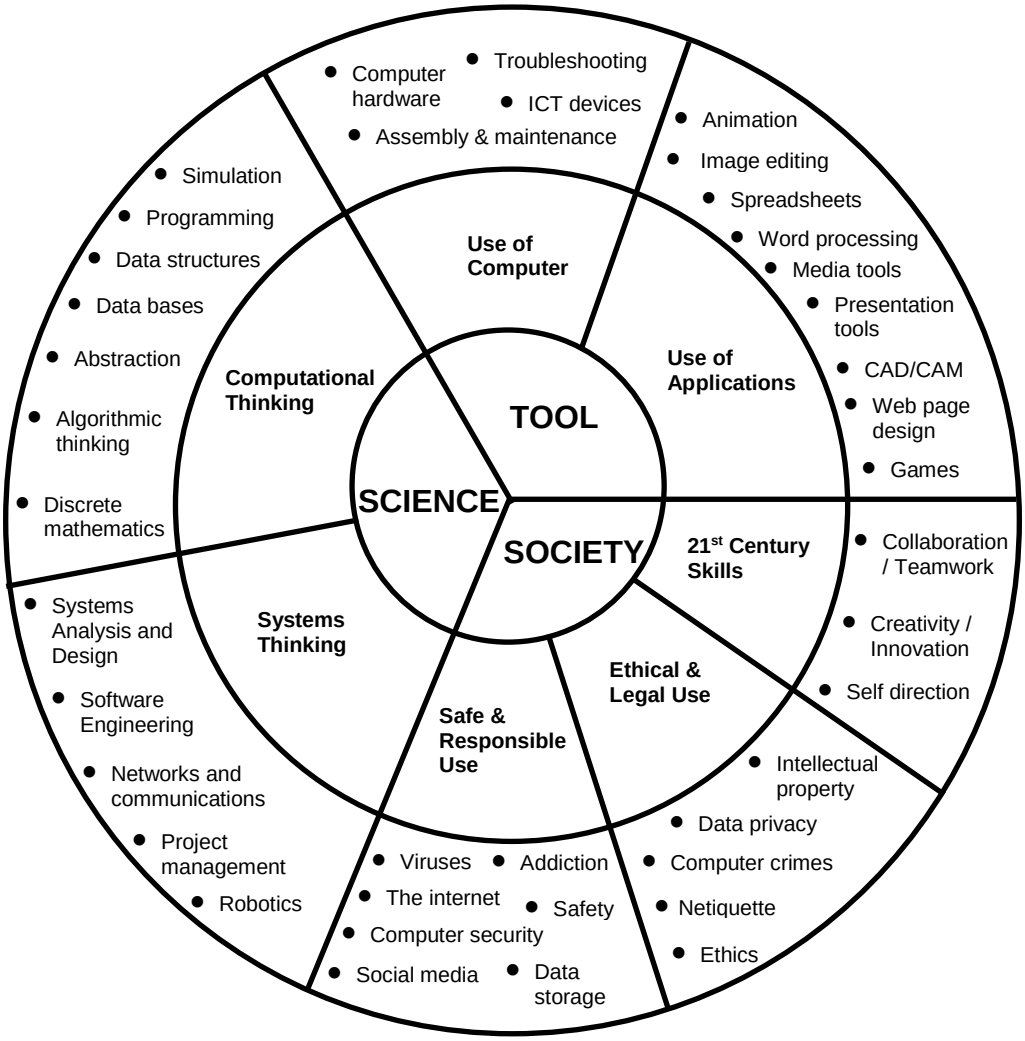

Figure 1: Computer Education Framework (The outer ring are examples of topics linked to each of the three dimensions shown in the inner ring. The middle ring expresses the desired concepts, skills and attitudes developed under each dimension.)

Computer in Society

This dimension focuses mainly on the ethical, legal and security issues relating to the use of computers and ICT in society. Issues commonly associated with this dimension include internet security, intellectual property, computer addiction, and data privacy.

The inclusion of the 21st century skills component reflects the impact of technology on the kind of skills needed at the workplace of the future. 21st century skills relevant to the ICT area include the ability to work collaboratively, produce creative work and be self-directed in learning.

1.5 Aims of Syllabus

The H2 Computing curriculum provides students with a broad understanding of the fundamental concepts and principles of computing, and a systemic understanding of how hardware and software work together in computing solutions. The syllabus covers the application of data structures and algorithms to process data through computer programs that are specifically designed and developed to solve authentic problems. Students will be exposed to authentic real-world contexts through hands-on practical assignments and projects in the realisation of computer programs from algorithms. Students will learn algorithm design and programming skills as a critical element of developing higher-order thinking skills. Through these learning experiences, the subject will provide a broad-based foundation for further studies in computing and other related fields.

The aims of the syllabus are to:

a) Acquire knowledge and understanding of core areas in computing covering concepts of algorithms, data structures, programming, databases and networks.

b) Develop and apply problem-solving and computational thinking skills to solve real-world problems using suitable algorithms and data structures in a web-based environment using a personal computer.

c) Develop (i) an appreciation of computing as a dynamic and creative field including awareness of recent developments in computer systems; and (ii) an understanding of the social, ethical, legal and economic implications of computing.

d) 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.

1.6 21st Century Competencies (21CC) in H2 Computing

The H2 Computing curriculum provides multiple opportunities for the development of 21st century competencies (21CC). For example, when a student designs and creates a web-based application to automate a task or improve a workflow, such as a task reminder or resource management application, they will develop inventive thinking when brainstorming for ideas. They will apply computational thinking in analysing how the application can be decomposed into modules that can be developed separately and integrated together later on. They will also apply formal reasoning when designing algorithms to solve the problem. As they test and improvise the design, they develop critical thinking during the refinement process. At the same time, students will have opportunities for collaborative learning and to hone their communication skills during presentations of their solutions.

The H2 curriculum provides opportunities for the development of 21CC in the areas of Communication, Collaboration and Information (CCI), Critical and Inventive Thinking (CIT) and Civic Literacy, Global Awareness and Cross-Cultural Skills (CGC). In addition, desired attitudes

such as creativity and resilience are fostered through the curriculum. These are illustrated in **Table A1**.

Table A1: Development of 21CC in A-Level H2 Computing

Computer Science Competencies and Attitudes	**21st Century Competencies Benchmarks (By end of JC2)**
	Critical and Inventive Thinking (CIT)
Ability to explore different plausible solutions and brainstorm ideas for problems (creativity).	1.1d: The student is able to generate ideas and explore different pathways that lead to solutions.
Ability to apply computational thinking by: • synthesizing knowledge and skills from the five core Computer Science areas; and • applying formal reasoning and systems thinking in the analysis, design and implementation of computer solutions.	2.1d: The student is able to use evidence and adopt different viewpoints to explain his/ her reasoning and decisions, having considered the implications of the relationship among different viewpoints.
Ability to debug and refine computer programs (computational thinking).	2.2d: The student is able to suspend judgment, reassess conclusions and consider alternatives to refine his/ her thoughts, attitudes, behaviour and actions.
Ability to: • analyse and simplify problems into manageable tasks (analytical thinking); • persist in developing computer solutions and debugging of programs when they do not work until the 'bugs' have been fixed (resilience); and • evaluate solutions using test cases (evaluative thinking).	3.1d: The student is able to identify essential elements of complex tasks, stay focused on them, take on diverse roles and persevere when they encounter difficulties and unexpected challenges. 3.2d: The student is able to manage uncertainty and adapt to diverse demands and challenges in new and unfamiliar contexts.
Computer Science Competencies and Attitudes	**21st Century Competencies Benchmarks (By end of JC2)**
	Communication, Collaboration and Information (CCI)
Ability to explain and communicate programming solutions and choice of data structures.	1.1e: The student is able to convey complex information and ideas coherently and clearly to influence and create impact for specific purposes and contexts.

	1.2e: The student is able to interact with others to construct and critically evaluate knowledge, new understanding and ideas.
Ability to be resourceful in searching and gathering pertinent information required to solve the computer-based problem.	2.1c: The student is able to refine search results, organise information systematically and manage information sensitively, while abiding by copyright regulations and minimising security risks in the handling of information. 2.2c: The student is able to verify the accuracy, credibility and currency of information across multiple sources.
Computer Science Competencies and Attitudes	**21st Century Competencies Benchmarks (By end of JC2)**
	Civic Literacy, Global Awareness and Cross-cultural Skills (CGC)
Ability to understand standards in ICT and the importance of standards to ensure environmental, health and public safety.	1.1e: The student is able to discuss issues that affect the culture, socio-economic development, governance, future and identity of Singapore and consider their implications. 2.1e: The student is able to analyse global trends and their implications for Singapore and other countries.
Adopt ethical practices and conduct of an ICT professional.	

SECTION 2: CONTENT

Syllabus Overview
Section 1: Algorithms & Data Structures
Section 2: Programming
Section 3: Data & Information
Section 4: Computer Networks

2. CONTENT

2.1 Syllabus Overview

This syllabus consists of four sections: (I) Algorithms and Data Structures, (II) Programming, (III) Data and Information, and (IV) Computer Networks that will cover common areas of fundamental computing concepts and theories to be undertaken at the pre-university level for two years.

The two sections of (I) Algorithms and Data Structures, and (II) Programming are considered as enduring concepts and skills which form the core fundamentals in computing courses while the teaching of the other two sections: (III) Data and information, and (IV) Computer Networks are considered as timely concepts and skills, intended to keep students abreast of new trends and developments in computing and technology.

The four sections and the respective units of study for each section are listed with details in subsequent pages.

Section 1: Algorithms and Data Structures

This section introduces the implementation of data structures to store and retrieve data efficiently, as well as their associated algorithms with the aim of developing problem solving skills. It also includes important concepts of decomposition and modularity, as well as techniques such as the use of decision tables to test algorithms. Students will need to abstract both data and procedures when they apply computational thinking to a problem. In addition, students will learn to implement various search and sort algorithms, and compare their efficiency for evaluation purposes. There are three units of study:

1.1 Algorithmic Representation
1.2 Fundamental Algorithms
1.3 Data Structures

1.1 Algorithmic Representation ***Write algorithms in pseudo-code and flowchart for given problems.***	
Ref	**Learning Outcome**
1.1.1	Use appropriate techniques or tools such as pseudo-code and flowchart to show program flow.
1.1.2	Use standard flowchart symbols.
1.1.3	Use a combination of various control structures.
1.1.4	Use decision tables to explore the actions for combinations of different input conditions. *Note: up to three conditions*
1.1.5	Use modular design to decompose a problem into smaller problems.

1.2 Fundamental Algorithms ***Understand algorithms for sorting and searching methods such as insertion sort, bubble sort, quicksort, merge sort, linear search, binary search and hash table search; and use examples to explain these methods.***	
Ref	**Learning Outcome**
1.2.1	Implement sort algorithms. - Insertion sort - Bubble sort - Quicksort - Merge sort
1.2.2	Use examples to explain sort algorithms.
1.2.3	Implement search algorithms. - Linear search - Binary search - Hash table search
1.2.4	Use examples to explain search algorithms.
1.2.5	Compare and describe the efficiencies of the sort and search algorithms using Big-O notation for time complexity (worst case). *Exclude: space complexity*

1.3 Data Structures ***Understand concept and write algorithms for stack and queue (linear and circular), linear linked list and binary search tree.***	
Ref	**Learning Outcome**
1.3.1	Understand the concept of static allocation of memory.
1.3.2	Understand the concept of dynamic allocation of memory.
1.3.3	Create, insert, and delete operations for stack and queue (linear and circular).
1.3.4	Understand the concept of free space list (which could be another linked list or an array).
1.3.5	Create, update (edit, insert, delete) and search operations for a linear linked list. *Exclude: doubly-linked list and circular linked list*
1.3.6	Create, update (edit, insert, *delete**) and search operations for a binary search tree. **Exclude: deletion of nodes from binary search tree*
1.3.7	Understand pre-order, in-order and post-order tree traversals; and application of in-order tree traversal for binary search tree.

Section 2: Programming

This section introduces students to the fundamental principles of programming in textual languages. Students will learn the common standards of programming style, programming constructs and library functions to be able to develop their own programs so as to solve a variety of problems. They are also required to write code to implement data structures such as stacks, queues, linked lists and binary search trees. In addition, students will also be expected to design, test and debug their own programs through lab-based practical assignments to ensure that they can work. The fundamental concepts of encapsulation, inheritance and polymorphism associated with object-oriented programming are also covered in this section. There are five units of study:

2.1 Coding Standards
2.2 Programming Elements and Constructs
2.3 Implementing Algorithms and Data Structures
2.4 Data Validation and Program Testing
2.5 Fundamentals of Object-Oriented Programming

2.1 Coding Standards ***Use common coding standards for programming style (which is dependent on programming language used).***	
Ref	**Learning Outcome**
2.1.1	Use indentation and white space.
2.1.2	Use naming conventions (e.g. meaningful identifier names)
2.1.3	Write comments (name of programmer, date written, program description and version book-keeping/control)

2.2 Programming Elements and Constructs ***Use programming language elements and constructs to write recursive and non-recursive programs to solve a variety of problems.***	
Ref	**Learning Outcome**
2.2.1	Understand the different data types: integer, real, char, string and Boolean; and initialise arrays (1-dimensional and 2-dimensional).
2.2.2	Use common library functions for input/output, strings and mathematical operations.
2.2.3	Apply the fundamental programming constructs to control the flow of program execution: - Sequence - Selection - Iteration
2.2.4	Use functions and procedures to modularise problem into chunks of code.
2.2.5	Understand the concept of recursion.
2.2.6	Trace the steps and list the results of recursive and non-recursive programs.
2.2.7	Understand the use of stacks in recursive programming.

2.3 Implementing Algorithms and Data Structures ***Use programming language elements and constructs to implement sort and search algorithms such as insertion sort, bubble sort, quicksort, merge sort, linear search, binary search, and hash table search, as well as data structures such as stacks, queues, linear linked lists and binary search trees.***	
Ref	**Learning Outcome**
2.3.1	Implement sort programs. - Insertion sort - Bubble sort - Quicksort - Merge sort
2.3.2	Implement search programs. - Linear search - Binary search - Hash table search
2.3.3	Write programs to implement operations for stacks, queues (linear and circular), linear linked lists and binary search trees. *Exclude: doubly-linked list and circular linked list*
2.3.4	Store data in and retrieve data from serial and sequential text files.

2.4 Data Validation and Program Testing ***Use data validation techniques and design test cases.***	
Ref	**Learning Outcome**
2.4.1	Explain the difference between data validation and data verification.
2.4.2	Understand data validation techniques such as: - range check - format check - length check - presence check - check digit
2.4.3	Identify, explain and correct syntax, logic and runtime errors.
2.4.4	Design appropriate test cases using normal, abnormal and extreme data for testing and debugging programs.

2.5 Fundamentals of Object-Oriented Programming ***Understand concepts of encapsulation, inheritance and polymorphism.***	
Ref	**Learning Outcome**
2.5.1	Define and understand classes and objects.
2.5.2	Understand encapsulation and how classes support information hiding and implementation independence.
2.5.3	Understand inheritance and how it promotes software reuse.
2.5.4	Understand polymorphism and how it enables code generalisation. *Exclude: method overloading and multiple inheritance*

Section 3: Data and Information

This section introduces students to the design, use and application of database management systems. The topics include relational data model, relational query languages and conceptual data design and modelling for relational database design. Students are expected to write programs to retrieve data from either a relational or non-relational database, process the data and return the processed data as a result. The use of databases also highlights the importance of data privacy and integrity. Students should be able to describe measures to safeguard the use of data. In addition, students should also be able to describe the code of conduct of a computing professional and discuss the social, economic and ethical implications of computing and technology. There are four units of study:

3.1 Data Representation
3.2 Character Encoding
3.3 Databases and Data Management
3.4 Social, Ethical, Legal and Economic Issues

3.1 Data Representation ***Understand that values can be represented in different number bases: denary, binary and hexadecimal.***	
Ref	**Learning Outcome**
3.1.1	Represent data in binary and hexadecimal forms.
3.1.2	Write programs to perform the conversion of positive integers between different number bases: denary, binary and hexadecimal forms; and display results.

3.2 Character Encoding ***Understand the use of ASCII code and Unicode to represent characters.***	
Ref	**Learning Outcome**
3.2.1	Give examples of where or how Unicode is used.
3.2.2	Use ASCII code in programs.

3.3 Databases and Data Management ***Understand, create and use SQL and NoSQL databases, as well as understand techniques to protect the privacy and integrity of data.***	
Ref	**Learning Outcome**
3.3.1	Determine the attributes of a database: table, record and field.
3.3.2	Explain the purpose of and use primary, secondary, composite and foreign keys in tables.
3.3.3	Explain with examples, the concept of data redundancy and data dependency.
3.3.4	Reduce data redundancy to third normal form (3NF).
3.3.5	Draw entity-relationship (ER) diagrams to show the relationship between tables.
3.3.6	Understand how NoSQL database management system addresses the shortcomings of relational database management system (SQL).
3.3.7	Explain the applications of SQL and NoSQL.
3.3.8	Use a programming language to work with both SQL and NoSQL databases.
3.3.9	Understand the need for privacy and integrity of data.
3.3.10	Describe methods to protect data.
3.3.11	Explain the difference between backup and archive.
3.3.12	Describe the need for version control and naming convention.
3.3.13	Explain how data in Singapore is protected under the Personal Data Protection Act to govern the collection, use and disclosure of personal data.

3.4 Social, Ethical, Legal and Economic Issues ***Understand the importance of ethics in the conduct of Computing professionals and the impact of Computing in different real-life situations.***	
Ref	**Learning Outcome**
3.4.1	Understand the code of ethics (conduct) of a Computing professional.
3.4.2	Describe the impact of computing on lifestyle and workplace for social and economic developments.
3.4.3	Discuss the social, ethical, legal and economic issues of computing and technology.

Section 4: Computer Networks

This section provides a broad view of the different types of basic networks, communication protocols and standards in a network. Students will be expected to understand concepts and techniques for developing web applications, describe the different types of threats to network security and propose mechanisms to protect and secure access to networks. They need to design, develop and test web applications as a consolidation of knowledge and skills through hands-on practical work and projects. There are three units of study:

4.1 Fundamentals of Computer Networks
4.2 Web Applications
4.3 Network Security

4.1	**Fundamentals of Computer Networks** ***Understand computer network technology.***
Ref	**Learning Outcome**
4.1.1	Explain the concepts of LAN, WAN, intranet and the structure of the internet.
4.1.2	Understand the concepts of IP addressing and domain name server (DNS).
4.1.3	Explain the need for communication protocols in a network.
4.1.4	Explain how data is transmitted in a packet-switching network.
4.1.5	Explain client-server architecture.
4.1.6	Implement an iterative server with socket programming. Given the server code, students should be able to implement the client code for a given scenario, and vice-versa e.g. for a tic-tac-toe game.

4.2	**Web Applications** ***Understand the concepts and techniques for developing web applications.***
Ref	**Learning Outcome**
4.2.1	Describe the differences between web applications and native applications.
4.2.2	State and apply usability principles in the design of web applications.
4.2.3	Use HTML, CSS (for clients) and Python (for the server) to create a web application that is able to: - accept user input (text and image file uploads) - process the input on the local server - store and retrieve data - display the output (as formatted text/ images/table)
4.2.4	Test a web application on a local server.

4.3	**Network Security** ***Understand computer network security in terms of threats, protection mechanisms and secure access.***
Ref	**Learning Outcome**
4.3.1	Understand how malware (e.g. worms and viruses) and denial of service (DOS) attacks can compromise computer systems.
4.3.2	Understand how firewall (filtering function), intrusion detection system (IDS) and intrusion prevention system (IPS) can be used to restrict network access; and their limitations.
4.3.3	Understand how encryption, digital signature, and authentication can ensure security of network applications.

SECTION 3: PEDAGOGY

Pedagogical Considerations
Pedagogical Approaches

3. PEDAGOGY

3.1 Pedagogical Considerations

Some of the factors that have shaped pedagogical approaches relevant to the teaching of Computing are identified as follows:

1. Alignment with Principles of Applied Learning

Applied learning is a notion that is gathering momentum in many educational contexts around the world, and is often equated to 'hands on' or practical learning experiences. Despite the many definitions of what constitutes applied learning, a number of recurring themes can be found across these definitions. These themes can be viewed as the pedagogical principles that support applied learning:

- emphasises the relevance of what is being learnt to the 'real world' outside the classroom, and makes that connection in an immediate and explicit manner;
- requires students to use hands-on or experiential learning to enact authentic scenarios, where students focus on learning and applying the skills and knowledge they need to solve a problem and implement a project;
- involves students and teachers in partnerships with the industries, community, institutions of higher learning, professional training bodies, and individuals outside school

H2 Computing aims to embody the principles of applied learning through a contextualized approach that would motivate students, while providing varied and enriched opportunities for the development of core skills and knowledge required for further education and active participation in their communities. Hence, teachers should plan their teaching to offer learning through doing experiences for their students to make sense of their "doing" by making connections to what they have learnt and constructing new knowledge for themselves in meaningful and purposeful ways so as to see the usefulness of Computing artefacts as products of innovations that solve real-world problems at homes, in their communities and the world at large. They should also plan lessons that provide opportunities for students to explore through the use of robots, micro-computers, computer simulations and visualization to deepen their own understanding and concretise abstract concepts.

2. Authentic Learning Experiences

Authentic learning typically focuses on real-world, complex problems and their solutions through a variety of problem-based activities, role-playing exercises, case studies, and participation in virtual communities of practice.

In "Authentic Learning for the 21st Century: An Overview", Marilyn Lombardi, illustrated that learning by doing is generally considered the most effective way to learn. She also espoused that the Internet and a variety of emerging communication, visualisation, and simulation

technologies make it possible to offer students authentic learning experiences ranging from experimentation to real-world problem solving.

The essence of authentic learning experience applicable to Computing is largely distilled in these design elements:

(i) **Real-world relevance**
The provision of authentic activities in the teaching of Computing should mirror the real-world tasks of professionals in practice as nearly as possible. Learning rises to the level of authenticity when students are required to work actively with abstract concepts, facts, and principles to create a web or mobile application or design a simple home network.

(ii) **Ill-defined problem**
Computing challenges are not easily solvable by the application of an existing algorithm. Instead, students will embark on authentic activities that are undefined and open to multiple interpretations, where they are required to identify for themselves the tasks and subtasks needed to complete the major task.

(iii) **Sustained investigation**
The solutions to computational problems cannot be solved in a matter of minutes or even hours. Instead, students are required to investigate in complex tasks over a sustained period of time despite experiencing initial frustration or disorientation.

(iv) **Multiple sources and perspectives**
Students would examine the task from a variety of theoretical and practical perspectives, using a variety of resources, and they will develop critical information skills of distinguishing relevant from irrelevant information in the process.

(v) **Collaboration**
Success is not achievable by an individual working alone. Authentic activities make collaboration integral to the task, both within the course and in the real world.

(vi) **Reflection (metacognition)**
In designing a solution, students have to deliberate over their choices and reflect on their learning, both individually and as a team or community.

(vii) **Multiple interpretations and outcomes**
Rather than yielding a single correct answer obtained by the application of theories and concepts, there may be diverse interpretations and varied solutions to a single computational problem.

Professionals with Computing skills often possess attributes and traits mentioned above as they work to design, develop and test programs or systems that can solve real-world problems. Getting students to solve real-world problems provides students with a full experience of this problem-solving process and the necessary opportunities to develop these useful attributes and traits.

3.2 Pedagogical Approaches

In line with principles of applied learning, the central pedagogical approach adopted is "learning through doing". This exploratory and hands-on approach is learner-centric, by encouraging learners to actively do something in order to concretise concepts and theories learnt. This not only allows students to directly observe and understand what they have learnt, but also helps them to develop a range of higher-order thinking skills. In addition, it encourages students to be self-directed and resourceful, which builds their confidence and self-management skills.

SECTION 4: ASSESSMENT

Assessment Philosophy
School-based Assessment
National Examination

4. ASSESSMENT

4.1 Assessment Philosophy

Assessment is an integral part of the learning process, and must be closely aligned with curricular objectives, content and pedagogy. Both school-based assessment and national examinations play important and different roles in our education system. A balanced assessment system should comprise of both assessment of learning as well as assessment for learning. Whether implemented in the classrooms or as part of national examinations, assessment should lead to meaningful learning. The 'what' and 'how' of assessment should be anchored on the clarity of purpose ('why'). There should be regular gathering of quantitative and qualitative information about a learner's progress and development, and such information should be used to inform learning and shape future teaching and learning practices.

The measurement of the efficacy of learning is dependent upon the gathering of accurate evidence about what students have learned for both teachers and students to make informed decisions about what to do next in order to improve student attainment. Assessment is integral to checking if learning has taken place as intended and planned.

The following are the three key messages of our assessment philosophy:

1. Assessment is integral to the learning process

Assessment is an iterative and continuous process which motivates learning and helps learners to achieve the learning outcomes stated in our curricular documents. The gathering and use of assessment information must become part of the ongoing learning process. Assessment can take the form of projects, classroom tests or national examinations, but the underlying goal should be to facilitate meaningful learning where the learning process is developmentally appropriate, caters to students' varied needs, and helps learners achieve our Desired Outcomes of Education (DOE).

2. Assessment begins with clarity of purpose

Assessment should be fit for purpose and based on sound educational principles. Decisions on 'what' to assess and 'how' to assess should be aligned with a clear purpose. A balanced assessment system consists of both assessments for learning as well as assessment of learning. In particular, formative assessment should be carried out during the instructional process for the purpose of improving teaching and learning, while summative assessment serves to provide information on students' mastery of content knowledge and skills.

3. Assessment should gather information to inform future practices

Assessment in schools should produce both quantitative and qualitative descriptions of learner performance to provide feedback for improving future teaching, learning and performance. Assessment should also help students become self-directed learners. There is

also the need to use different modes of assessment so that we can determine how best to support students in their progress with respect to different domains of learning.

4.2 School-based Assessment

Assessment is an important part of classroom teaching and learning and is an ongoing process by which teachers gather information about students' learning to inform and support teaching. Assessment generally serves two purposes:

- Formative Assessment, such as projects, are used to determine how students are progressing through certain learning outcomes during a series of learning activities or to establish prior knowledge at the start of the learning cycle. This type of assessment can be used to identify learning gaps to provide timely feedback to students on their learning, and inform teachers on planning for future instruction.

- Summative Assessment, such as tests, school and national examinations, are used at the end of a series of learning activities to determine the level of students' attainment of the desired learning outcomes. It is commonly used for placement and grading.

In Computing, students are expected to apply their theoretical knowledge explicating computational thinking to solve real-life problems. To develop the necessary skills, teachers need to gather timely and regular information on their students' level of competency so as to provide targeted feedback to help students improve. The adopted pedagogies and resources offer opportunities for assessment for learning by getting the students to perform tasks that demonstrate their understanding of the concepts.

4.3 National Examination

Details on the national examination are found in the Examination Syllabus on the website of the Singapore Examinations and Assessment Board.