O-LEVEL COMPUTING SYLLABUS

Upper Secondary

Implementation starting with 2017 Secondary Three Cohort

Ministry of Education
SINGAPORE

CONTENTS

SECTION 1: INTRODUCTION

1. INTRODUCTION

General Overview

Over the past decades, advancement in computer technology has been the key driving force behind much of the changes witnessed in many aspects of our lives. Beyond this impact on our daily lives as users of technology, Computing has become so entrenched in the fields of Science, Technologies, Engineering and Mathematics (STEM) that it has transformed the very practices of those disciplines.

Data analytics are transforming the future of business and research in diverse areas. We will need stronger computational capacity and people with considerable computational thinking skills to automate and streamline processes, work with data (e.g. text and images) to discern trends and draw inferences, develop hypotheses and innovate in fields like medicine, science and engineering to solve complex problems.

Professionals in finance, retail and manufacturing will require computational thinking skills to be competitive in their fields. For example, it will be an asset to acquire the ability to use a computer to analyse data to predict market and business trends, understand economic conditions and build participatory relationships with clients in the digital environment. Students who study O-level Computing would acquire a stronger foundation in computational thinking that would be useful to their further studies and careers.

Purpose and Philosophy

The global computing education landscape is experiencing a renewed emphasis on programming (coding) and the fundamental computational skillsets that will prepare students to thrive in a more digitally-connected world and work place. The plethora of applications on mobile and computing devices have changed the way we live, learn and work. The impact is felt across all age groups. To narrow the digital divide, various government schemes and initiatives were launched to reach out to the population so that more people can harness technology effectively to improve their lives and contribute to society.

While the current baseline ICT programme equips all students with basic computer literacy skills to be informed users of technology, the O-level Computing curriculum aims to grow students' interest and competency in more advanced concepts and skills. This will equip them with the necessary foundation to continue with post-secondary computing courses. A secondary aim is to encourage students to consider careers in computing technology and systems or as skilled programmers, system developers and software engineers. Our aspiration is for this group of students with the passion for Computing to eventually harness their talent to solve complex problems or create new value propositions in society through technology.

Importance of Subject

The value of computing arises from the fact that it has the ability to integrate the use of software and hardware to create new artefacts, solve/address real-world problems and perform tasks. Computing is grounded in (i) computational thinking and (ii) systems thinking; and enables (iii) exploration and innovation. Thus, *Computational Thinking* and *Systems Thinking* form the two arms in the dimension of Computer as a Science in our computer education framework (shown in Figure 1.1). The other two dimensions on Computer as a Tool and Computer in Society also cover important areas in Computing and are taught so that students are able to develop their ideas and creativity through the use of information and communication technology (ICT) to contribute to society.

Computational Thinking

Computational thinking is a thought process that involves formal reasoning, logical and algorithmic thinking, and the reformulation of a problem so that a computer-based solution is viable.

In the O-level Computing subject, students are taught how to go through a systematic process of thinking when solving problems (abstraction), formulating steps for solutions (algorithmic thinking) and writing computer programs (programming/coding) to produce the solutions. Hence, students develop computational thinking skills when analysing problem situations and reformulating them into problems with computational solutions. They also develop computational thinking when coding effective and efficient programming solutions. Students with computational thinking skills will be able to apply their skills across other subject disciplines and be better enabled to solve real-world problems in those disciplines. Armed with computational thinking skills, our students will have a competitive edge in the increasingly digital landscape.

The promotion of computational thinking in students also featured predominantly in the syllabuses offered in the countries scanned (e.g. UK, US, Australia and Hong Kong). Computational thinking and the study of Computing allows the students to understand the digital world in a deeper way; just as physics allows students to understand the physical world better and the study of biology allows a better understanding of the biological world.

Computer Education Framework

The computer education framework comprises three dimensions:

- 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[1], is a way people solve problems and that it is not about trying to get people to think like computers[2]. 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 computer scientists.

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

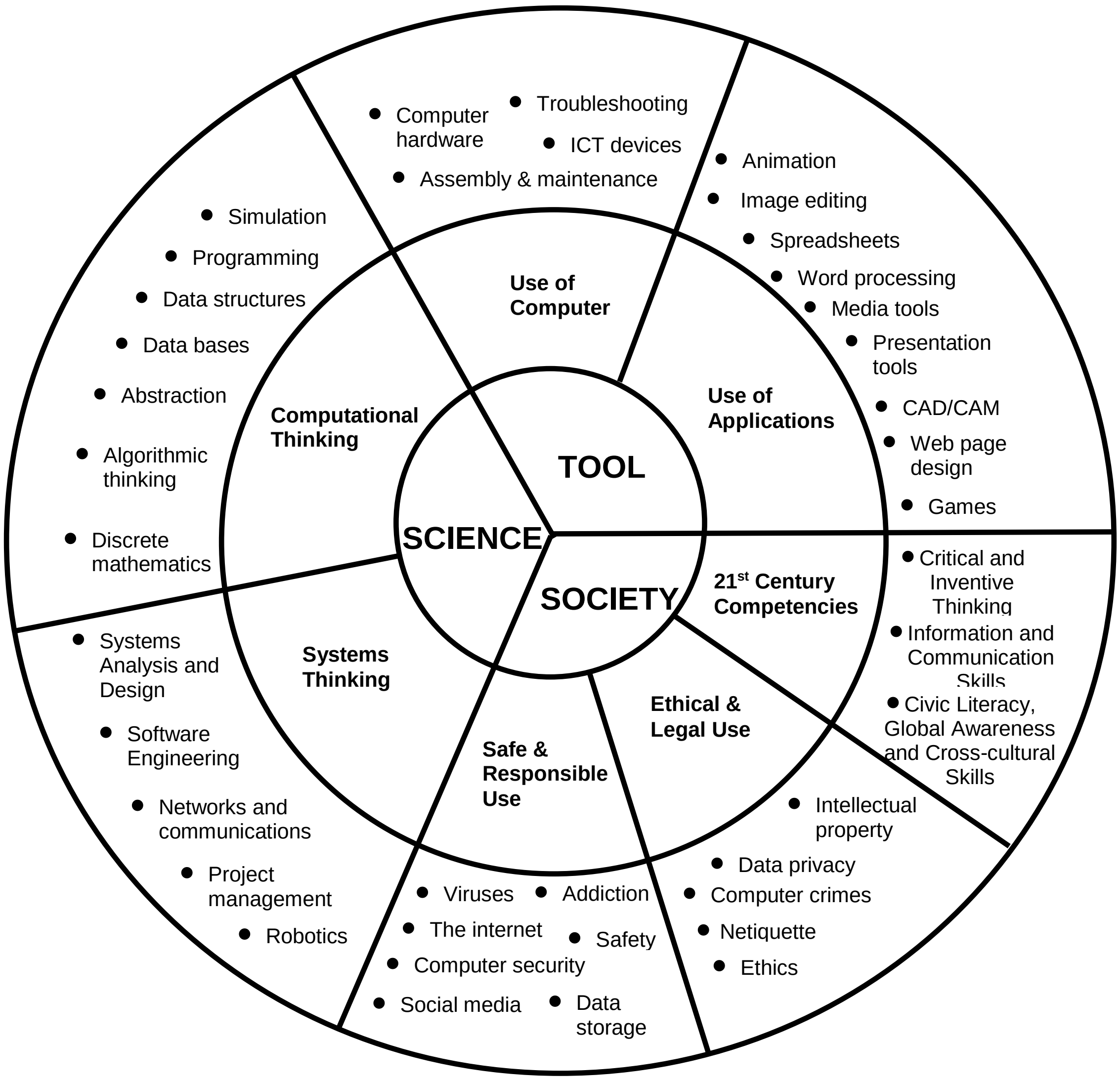


Figure 1.1: Computer Education Framework (The outer ring are examples of topics that could be studied for the three dimensions shown in the inner ring. The middle ring shows the components of each dimension.)

[1] Jeannette M. Wing is the President's Professor of Computer Science and head of the Computer Science Department at Carnegie Mellon University.
[2] J. M. Wing. Computational Thinking. Communications of the Association for Computing Machinery (ACM), March 2006, Vol. 49(3).

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.

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

Big Ideas in Computing

The US College Board and the National Science Foundation (2010)[3] focused on practices of computational thinking for high school CS curricula based on seven 'big ideas' of Computing. The 'big ideas' are:

(a) Computing is a creative human activity.
(b) Abstraction reduces information and detail to focus on concepts relevant to understanding and solving problems.
(c) Data and information facilitates the creation of knowledge.
(d) Algorithms are tools for developing and expressing solutions to computational problems.
(e) Programming is a creative process that produces computational artefacts.
(f) Digital devices, systems, and the networks that interconnect them enable and foster computational approaches to solving problems.
(g) Computing enables innovation in other fields, including science, social science, humanities, arts, medicine, engineering, and business.

In addition, Stanford University's[4] undergraduate program on 'Ways of Thinking/Ways of Doing' describes formal reasoning as a way of thinking that involves rigorous deductive (logical and analytical) thinking. This way of thinking underpins decision making and analysis skills that are applied in many fields, including Computing.

[3] AP Computer Science: Principles – Big Ideas, Key Concepts and Supporting Concepts (2010, The College Board and National Science Foundation).

4 The study of undergraduate program at Stanford: Ways of Thinking/Ways of Doing, a capacity-based approach to fostering breadth in General Education Electives.

Syllabus Aims

The 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. The two-year course at upper secondary level is to enable students to:

1. Apply logical reasoning and algorithmic thinking[5] in analysing problem situations and developing solutions;
2. Develop simple programs through the use of appropriate programming language(s);
3. Understand how and where information communications technology (ICT)[6] is used in daily life;
4. Understand and explain the ethical, social and economic issues associated with the use of ICT.

Students will be taught how data can be managed and processed by using functions and formulas in a spreadsheet application to produce information and inform decision making; how digital systems work and how this knowledge can be used to create safe homes and secured networks; and how to use the computer efficiently and effectively to code solutions for problem situations. Details of what the students will be taught are in the following section.

[5] Algorithmic thinking is a way of getting to a solution through clear definition of the steps. Instead of a single answer, a set of instructions or rules is developed, that if followed precisely leads to answers.
[6] Information and Communications Technology (ICT) refers broadly to technology involving computing devices, software and other hardware.

SECTION 2: CONTENT

2. CONTENT

Syllabus Overview

This syllabus comprises four modules of study to cover five common areas of computer science concepts and skills. The study is undertaken at the upper secondary levels for two years. The four modules and the units of study for each module are as listed with details in subsequent pages. Schools are encouraged to have in place, at lower secondary levels, a computer science programme that gives students a firm foundation to undertake the subject at upper secondary levels.

The four modules are:

Module I – Data and Information
- Data Management
- Data Representation
- Ethical, Social and Economic Issues

Module II – Systems and Communications
- Computer Architecture
- Data Communications

Module III – Abstraction and Algorithms
- Problem Analysis
- Algorithm Design

Module IV – Programming
- Program Development
- Program Testing

Module I: Data and Information

This module is about the handling and processing of data in computer systems, as well as the need to be ethical when dealing with data. Students should be able to identify different types of data, understand and explain what the data is for, and explain how the data is represented or organised for processing and output, with reference to a given problem. Students will be more aware of ethical issues with respect to data, including privacy of data. There are three units of study in this module:

1.1 Data Management
1.2 Data Representation
1.3 Ethical, Social and Economic Issues

1.1 Data Management

This unit of study emphasises logical thinking and reasoning through data analysis, data processing and visual representations of data. Students will use the spread sheet application software in hands-on activities to enhance learning of functions and formulae to compute and process data.

Ref	**Learning outcome**
	Students should be able to
1.1.1	Tabulate data under appropriate column headings (i.e. data field names) and data types (e.g. numeric, text and date). *Exclude: plotting of charts*
1.1.2	Use mathematical, statistical and financial functions in appropriate contexts, including what-if data analysis, to • find the total of a list of numbers, average of a list of numbers, min/max of a list of numbers, square root of a number, simple interest, remainder after division of numbers; • round values; • randomise values; • convert data from one type to another (e.g. get integer values from decimal values); and • count number of data items. *The list of 41 examinable functions are:* TODAY, LEN, MID, LEFT, CEILING, FLOOR, MAX, MIN, IF, COUNT, COUNTA, COUNTIF, COUNTBLANK, POWER, SQRT, ROUND, RANDBETWEEN, RAND, SUM, SUMIF, MOD, MODE.SNGL, MEDIAN, AVERAGE, SMALL, LARGE, PERCENTILE, QUARTILE, STDEV, VAR, AND, OR, NOT, HLOOKUP, VLOOKUP, PV, FV, RATE, PMT, IPMT, PPMT. *Students should not be penalised for using equivalent functions not listed.*

Ref	**Learning outcome**
	Students should be able to
1.1.3	Understand and use conditional statements (simple and nested): COUNTIF and IF with relational and Boolean operators such as AND, NOT and OR. *<u>Include:</u> conditional formatting*
1.1.4	Use functions effectively to look up data in rows or columns (horizontal and vertical table lookups) in a list or table for data processing.

1.2 Data Representation

This unit of study explains how data is represented internally as binary numbers, how the conversion of data from one number system to another is done, and where these number systems are used. The number systems covered here are binary number system, decimal number system and hexadecimal number system.

Ref	**Learning outcome**
	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	Describe situations in which the number systems are used such as ASCII codes, IP (Internet Protocol) and Media Access Control (MAC) addresses, and RGB codes.

1.3 Ethical, Social and Economic Issues

This unit of study covers ethical use and security of data in desktop applications or applications on networks like the internet. Social and economic issues arising from the use of computing technologies in workplace settings are discussed.

Ref	**Learning outcome**
	Students should be able to
1.3.1	Understand how data can be kept safe from accidental damage due to data corruption or human errors and malicious actions such as unauthorised viewing, deleting, copying and corrupting or malware.
1.3.2	Understand effects of cyberattacks such as spam, spyware, cookies, phishing and pharming and use of safety measures such as appropriate hardware and software.
1.3.3	Describe ethical issues that relate to the use of computers, public/private networks, freeware, shareware and open courseware; and the sharing of information.
1.3.4	Compare the positive and negative social and economic impacts of technology in education, communication (e.g. via mobile apps and social media), finance, medicine/healthcare, transportation and entertainment; and explain plagiarism and software piracy.

Module II: Systems and Communications

This module is about systems involving computer technology and computing devices. Students will learn the inter-relationships between parts and whole of a system; as well as the functions of parts of systems in enabling communications between human and computing device (machine), machine and machine, and within a machine. There are two units of study:

2.1 Computer Architecture
2.2 Data Communications

2.1 Computer Architecture

This unit of study covers the basic computer architecture and components of a computer network. Students will understand the purpose of hardware and software required in a computer system or network but they are not required to know how these work technically. Students will have the opportunity to work with hardware components and understand the integrative use of hardware and software through learning by doing.

Ref	**Learning outcome**
	Students should be able to
2.1.1	Describe basic computer architecture with reference to - Computer processor - Memory - Data and address buses - Input (e.g. data and instructions) and output (e.g. intermediate and final results of processing) - External storage *Exclude:* *Fetch and execute cycle*
2.1.2	Recognise a logic gate from its truth table and evaluate Boolean statements by means of truth tables.
2.1.3	Construct the truth tables for given logic circuits (maximum 3 inputs).
2.1.4	Design and construct simple logic circuits using AND, OR, NOT, NAND and NOR.

2.2 Data Communications

This unit of study uses networks as the context for understanding sharing of resources and data communications. Students should have a general understanding of how data is transmitted and the need to check for data accuracy and data security in transmissions.

Ref	**Learning outcome**
	Students should be able to
2.2.1	Identify the different network devices: router, bridge and network interface card; and explain their functions.
2.2.2	Describe the difference between wired and wireless networks and explain the factors that will determine the use of each type of network. *Include: descriptions of LAN/MAN/WAN*
2.2.3	Describe the components for a simple home network and design a simple home network.
2.2.4	Compare and contrast client-server and peer-to-peer network strategies with emphasis on: - Purpose - Function - Organisation - Bandwidth *Include: topology principles of bus, ring and star networks*
2.2.5	Explain the use of parity and checksums in data transmission.

Module III: Abstraction and Algorithms

This module is about problem solving and how a problem may be solved by breaking it into smaller, manageable parts and solving all the smaller parts. An algorithm describes a solution for the problem that is independent of a programming language and may be presented in pseudo-code (where program structures will be more pronounced) or diagrammatically (flowchart). Students should be able to know the difference between pseudo-code and flowchart. There are two units of study:

3.1 Problem Analysis
3.2 Algorithm Design

3.1 Problem Analysis

This unit of study covers problem interpretation and analysis. Students will learn how to approach problem solving in a systematic manner. They will learn problem-solving strategies like breaking a problem into its parts and solving the problem by first solving the individual parts. They will have opportunities to reinforce their understanding through hands-on activities in solving simple real-world problems.

Ref	**Learning outcome**
	Students should be able to
3.1.1	Define and state a problem by • Identifying the required inputs • Identifying the required outputs • Stating the processes required Examples: • Business: produce an itemised list of items purchased, cost of each item, total cost payable (like a receipt) or calculate interest on mortgages and print instalments over a period of time • Education: find and print the student with the top score in each subject or check user inputs against expected input (like test scoring); find the mean subject grade (MSG) for a class • Scientific/Mathematics: determine whether an input number is odd or even and whether a number is divisible by another number • Entertainment: a number guessing game or any game involving text manipulation
3.1.2	Solve problems by decomposing[7] them into smaller and manageable parts.
3.1.3	Identify common elements across similar problems and state generalisations[8].

[7] Decomposition is a way of thinking about problems, algorithms, artefacts, processes and systems in terms of their parts. The parts can then be understood, solved, developed and evaluated separately. This approach makes it easier to solve complex problems and design large systems.

[8] Generalisation is a way of quickly solving new problems based on previous problems that had been solved. That is, the algorithm that works for a previous specific problem is adapted so that it solves a whole class of similar problems through the identification of patterns and commonalities in problems, processes, solutions, or data.

3.2 Algorithm Design

This unit of study is about interpreting and understanding algorithms; correcting and writing algorithms for given problems; as well as refinement of algorithms. Students' learning will be enhanced through written exercises, classroom discussions and presentations.

Ref	**Learning outcome**
	Students should be able to
3.2.1	Perform a dry run of a set of steps to determine its purpose and/or output.
3.2.2	Produce trace tables to show values of variables at each stage in a set of steps.
3.2.3	Locate logic errors in an algorithm, and correct or modify an algorithm to remove the errors or for changes in task specification.
3.2.4	Produce an algorithm, in pseudo-code or diagrammatically, to solve a problem.

Module IV: Programming

This module is about application and development of logical thinking and reasoning, as well as problem-solving skills through the design and development of software solutions using programming language(s). An algorithm describes a solution independent of a programming language while a programming language depicts the solution that is workable on a computing device. Students will get to test whether their algorithms work as planned when they run the programming solutions. There are two units of study:

4.1 Program Development
4.2 Program Testing

4.1 Program Development

This unit of study covers the development of programming solutions (coding) for simple problem situations. Students will reinforce their understanding through practical work. Python will be the recommended programming language.

Ref	**Learning outcome**
	Students should be able to
4.1.1	Understand and describe the stages in developing a program: gather requirements, plan solutions, write code, test and refine code, implement code.
4.1.2	Understand and use sequence, selection and iteration constructs to create a program.
4.1.3	Use and justify the use of variables, constants and simple lists (1-D array) in different problem contexts.
4.1.4	Understand, use and justify the use of different data types and built-in functions in programs.
4.1.5	Produce programming solution for a given problem to • find min/max value in a list • find average of a list of numeric values • search for an item in a list and report the result of the search • find check digits • find the length of a string of characters • extract required characters from a string of characters

4.2 Program Testing

This unit of study covers the testing and refinement of programs based on test results. Students will learn the appropriate use of test cases and what type of testing is necessary and/or sufficient. Students will reinforce their learning and understanding through hands-on practical work. Python will be the recommended programming language for coding.

Ref	**Learning outcome**
	Students should be able to
4.2.1	Identify and justify the use of data validation techniques.
4.2.2	Validate input data for acceptance by performing • length check, • range check, • presence check, and • format check
4.2.3	Design appropriate test cases to cover normal, error and boundary conditions, and specify what is being tested for each test case.
4.2.4	Understand and describe types of program errors: syntax, logic and run-time; and explain why they occur.
4.2.5	Explain how translators are used to detect syntax errors and state the difference between interpreter and compiler translators.
4.2.6	Understand and apply debugging techniques to isolate/identify and debug program errors: use of intermittent print statements, walking through a program, testing program in small chunks or by parts, and using built-in debugging help.

SECTION 3: PEDAGOGY

Applied Learning
Learn by Doing

3. PEDAGOGY

Applied Learning

As an applied subject, there is greater emphasis on learning by doing in the real-world context. The pedagogies and activities used must thus be suitable for applied learning. In addition, it is intended that students be provided with opportunities to acquire 21st Century Competencies through applied learning.

21st Century Competencies (21CC)

Table 3.1 illustrates how the O-Level Computing curriculum is aligned with the Standards and Benchmarks for 21CC. To illustrate, students will develop critical and inventive thinking skills (2.1d) when they are tasked to design a program to compute point-of-sales receipts. The students will need to assess information and process requirements, and analyse costs and benefits before applying sound reasoning and decision-making to propose a feasible solution with respect to the hardware and software to be acquired.

Table 3.1: Alignment with 21CC Standards and Benchmarks

O-level Computing Competencies and Attitudes	21st Century Competencies Benchmarks (By end of S4/S5)
Computer as a Science	**Critical and Inventive Thinking (CIT)**
Ability to brainstorm ideas for problem solutions.	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 judgement, reassess conclusions and consider alternatives to refine his/ her thoughts, attitudes, behaviour and actions.
Ability to:	3.1d: The student is able to identify essential elements of complex tasks, stay focused on them, take on diverse roles

• analyse and simplify problems into manageable tasks (analytical thinking); • persist in developing computer solutions for the problems (resilience); and • evaluate solutions using test cases (evaluative thinking).	and persevere when they encounter difficulties and unexpected challenges.
Computer as a Tool	**Information and Communication Skills (ICS)**
Ability to explain and communicate programming solutions.	1.2d: The student is able to use information and ideas developed collectively with others to create new information, products and/ or solutions.
Ability to be resourceful in searching for pertinent information required to solve the problem.	2.1d: The student is able to integrate information from a variety of sources to complete a task.
Computer in Society	**Civic Literacy, Global Awareness and Cross-cultural Skills (CGC)**
Ability to understand data protection, copyright issues and intellectual property rights.	1.1d: The student is able to discuss issues that affect the culture, socio-economic development, governance, future and identity of Singapore and use evidence to support their viewpoints.
Adopt ethical practices in cyberspace and social media.	

<u>Authentic Situations</u>

Real case studies on data protection, for example, will be used for students to inquire the need for data privacy and integrity, and to compare/contrast the different measures of data protection. Students will realise the dangers and understand the mistakes made by others through online transactions or sharing of information via the internet; and will learn to be discerning and careful with their personal information while adopting ethical practices in cyberspace.

Learn by Doing

A wide range of pedagogies is recommended for the teaching of O-Level Computing. For example, using a flipped classroom approach, the students may watch a video (at their own time) on the design considerations and actual setup of a simple wireless network in a home or business setting. The students will then set up a simple network physically in the computer classroom or use a web-based similar available in the Student Learning Space and learn by doing. While the pedagogies applied are dependent on the nature of the topic, a problem-driven approach could be used.

Section 4: Assessment

School-based Assessment
National Examination

4. ASSESSMENT

School-based Assessment

Assessment for Learning

Students are provided with opportunities for deep and enriched learning experience through the use of online learning environments in the Student Learning Space or the internet. They can monitor and self-regulate their pace and progress of learning. They receive immediate feedback from these learning environments on errors in their programming solutions and whether the outcomes of their programming solutions are as intended.

The six weeks set aside for the programming project will enable students to collaborate and learn from one another. Besides assessment for learning, assessment of learning also takes place when students apply programming concepts and skills that they have learnt to the project. Computational thinking would also be developed and if the project is done on a group basis, students would also learn to explore different pathways to solutions and enhance communication skills.

Assessment of Learning

The Student Learning Space will also allow students to assess their learning as well as learn from the assessment tasks. School-based assessment can also consist of timed written and practical assessment of students' understanding of the four modules in the syllabus. The questions should test more on students' understanding and application of concepts and skills learnt, and less on recall of information. The written paper may comprise short-structured questions of variable mark values.

A sample lab-based paper may consist of a spreadsheet task and one or two programming tasks. Alternatively, the paper may consist of a series of planning and progressively developed tasks culminating in a final programming solution.

The format of the assessment papers may also be modelled after the format of the national examinations. The marks for school-based assessment may be used for reporting students' performance at end of Semesters.

National Examination

Assessment Objectives
The examination will assess:

a) Knowledge and understanding of basic computing technology and systems, concepts, algorithms, techniques and tools;

b) Application of knowledge and understanding to analyse and solve computing problems;

c) Development, testing and refinement of solutions using appropriate software application(s) and/or programming language(s).

The weighting for each assessment objective is shown in Table 4.1.

Table 4.1 Specification Grid

Ref	Assessment Objective	Weighting in Paper 1	Weighting in Paper 2	Overall Weighting
(a)	Knowledge and understanding	~ 35%	~ 5%	40%
(b)	Application	~ 25%	~ 5%	30%
(c)	Development, testing and refinement	~10%	~20%	30%
	TOTAL	70%	30%	100%

Students can handle and process data in computer systems, as well as the need to be ethical when dealing with data. They will demonstrate problem-solving techniques through analysing and writing programming solutions for a range of computing problems in business, education, mathematics and science. Students will be able to demonstrate computational thinking through the design and development of programming solutions.

Mode of Examination

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

Paper 1 (Written examination, 2 hours)
This paper tests knowledge, understanding and application of concepts and skills in all the four modules. The questions consist of a mixture of

- short answer questions
- matching questions
- cloze passage
- structured questions

This paper carries 70% 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, spreadsheet and programming[9] software, tests Module 1 (Unit 1.1: Data Management) and Module 4 (Programming). Four structured questions will be set based on the following topics:

- Use of spreadsheet functions and features
- Refinement of program
- Debugging of program
- Development of program with no more than 40 lines

Development of program will carry 20 marks. The remaining three questions average 10 marks.

Candidates will submit soft copies of the required work for marking. The allotted time includes time for saving the required work in the candidates' computer. This paper carries 30% of the total marks, and is marked out of 50 marks.

Table 4.2 summarises the formats of the two compulsory papers.

Table 4.2: Format of Papers

Paper	Mode	Duration	Weighting	Marks	Format	Modules Assessed
1	Written	2 hours	70%	80	A mixture of • Short-answer questions • Matching questions • Cloze passage • Structured questions	All the four modules
2	Lab-based practical exam	2 hours 30 minutes	30%	50	4 compulsory structured questions on • Use of spreadsheet functions and features • Refinement of programme • Debugging of programme • Development of programme with no more than 40 lines of code Development of programme will carry 20 marks. The remaining three questions will carry an average of 10 marks per question.	Unit 1.1 Data Management from module 1 Module 4: Programming

[9] *The centre will be required to declare the programming language(s) to be used for the examination before the centre begins teaching the course for the cohort taking the examination.*

Use of Calculator

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

Glossary List
Candidates will be provided in the examination with a glossary list of programming syntax. The glossary is in Annex A.

Centre Infrastructure for Lab-based Examination
The Centre will ensure adequate hardware and software facilities to support the examination of its candidates for Paper 2, which will be administered over at most two shifts on the day of the examination. Each candidate should have the sole use of a personal computer for the purpose of the examination. Candidates should be able to access Spreadsheet application software and programming language software. The Centre will be required to declare the name and version number of the software to be used for the cohort sitting for the examination at least two years before the cohort sits for the examination.

Quick Reference for Python

This quick reference shows some examples of the Python language constructs. The complete Python language is not limited to these examples.

1. Identifiers

When naming functions, variables and modules, the following rules must be observed:

- Names should begin with character 'a' - 'z' or 'A' - 'Z' or '_'
  and followed by alphanumeric characters or '_' .
- Reserved words should not be used.
- User-defined identifiers are case sensitive.

2. Comments and Documentation Strings

```
# This is a comment

"""
   This is a documentation string
   over multiple lines
"""
```

3. Input/Output

```
print ("This is a string")

s = input ("Instructions to prompt for data entry.")
```

4. Import

import <module>

e.g. **import** math

5. Data Type

Data Type	Notes
int	integer
float	real number
bool	boolean
str	string (immutable)
list	series of values

6. Assignment

Assignment Statement	Notes
a = 1	integer
b = c	variable
d = "This is a string"	string
mylist = [1, 2, 3, 4, 5]	list or array

7. Arithmetic Operators

Operator	Notes
+ -	plus, subtract
* /	multiply, divide
%	remainder or modulus
**	exponential or power
//	quotient of the floor division

8. Relational Operators

Operator	Notes
==	equality
!=	not equal to
> >=	greater than, greater than or equal to
< <=	less than, less than or equal to

9. Boolean Expression

Boolean Expression	Notes
a and b	logical and
a or b	logical or
not a	logical not

10. Iteration

while loop	for loop

while condition(s):
    <statement(s)>
	for i **in** range(n):
	    <statement(s)>
	
	for record **in** records:
	    <statement(s)>

11. Selection

Type 1

```
if condition(s):
    <statement(s)>
```

Type 2

```
if condition(s):
    <statement(s)>
else:
    <statement(s)>
```

Type 3

```
if condition(s):
    <statement(s)>
elif condition(s):
    <statement(s)>
else:
    <statement(s)>
```

12. Built-in Functions

(a) Basic functions

abs( )	chr( )	float()	input( )	int( )
ord( )	print( )	range( )	round( )	str( )
format( )				

(b) Mathematical functions

ceil( )	exp( )	fabs( )	floor( )	log( )
max()	min()	pow( )	sqrt( )	trunc( )

(c) String functions

endswith()	find()	isalnum( )	isalpha( )	isdigit( )
islower( )	isspace( )	isupper( )	len()	lower( )
startswith()	upper( )			

13. Reserved Words

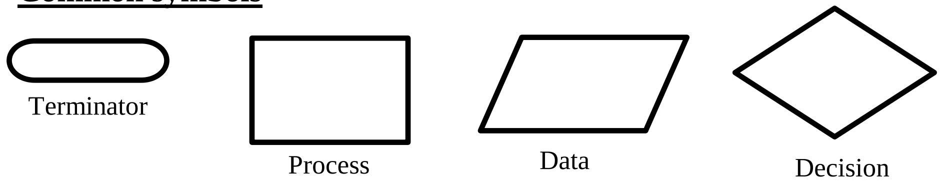
Reserved words cannot be used as identifiers. They are part of the syntax of the language.

False	None	True	and	as
assert	break	class	continue	def
del	elif	else	except	finally
for	from	global	If	import
in	is	lambda	nonlocal	not
or	pass	raise	return	try
while	with	yield		

Quick Reference for Flowcharting

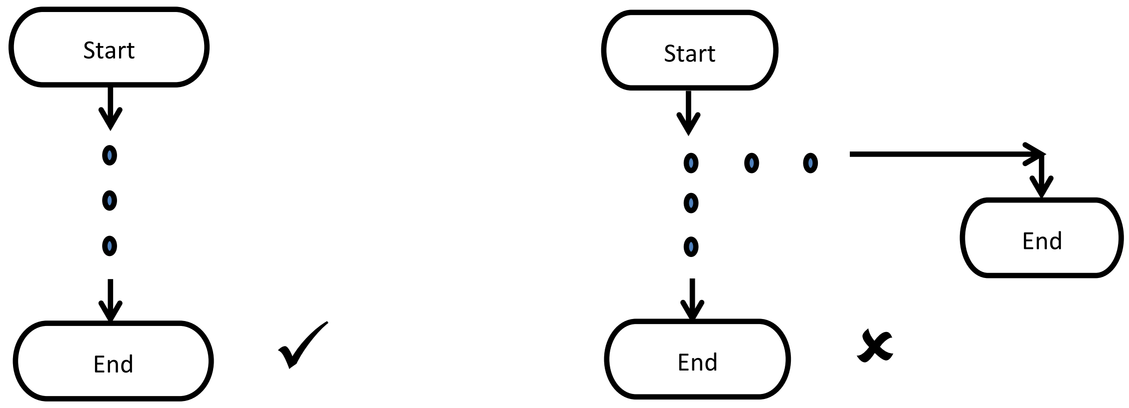
This quick reference shows four common symbols used in program flowcharts, and provide a standard guide for constructing program flowcharts.

1. Common symbols

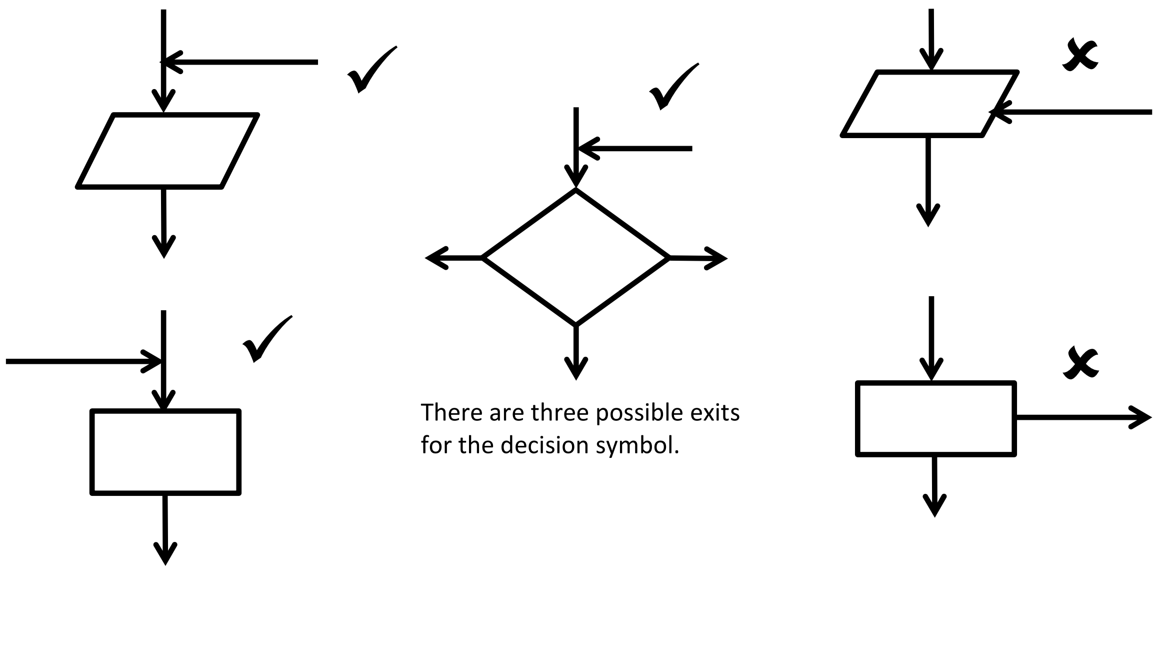


2. Constructing flowcharts

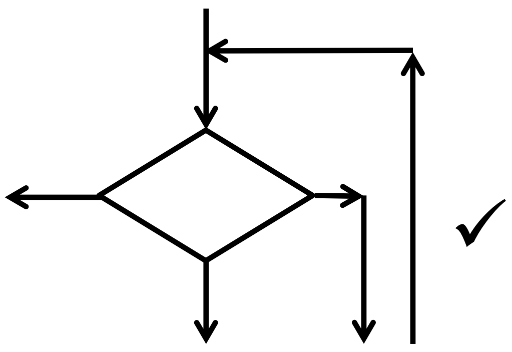
(a) Start and end with ONE terminator symbol

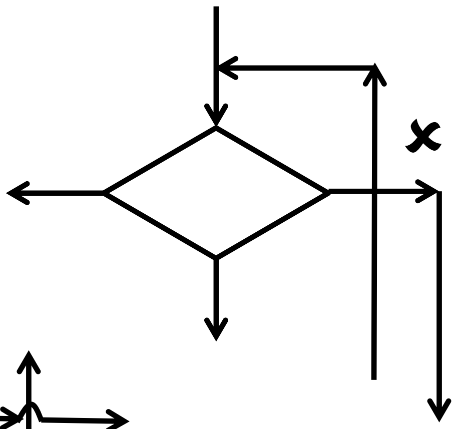


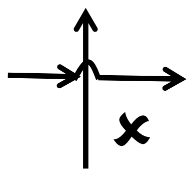
(b) Have a single entry and a single exit (except for the terminator and decision symbols)



(c) Flow lines should not cross one another







Some people used a kink in the flow line but this is not recommended for our students.

Glossary of Terms

Students should be familiar with the following terms in the learning outcomes, and these terms would be similarly used in the examination questions.

Term	Learning outcomes	Definition
Apply	4.2.6	Carry out or use common techniques or procedures.
Compare	1.3.4	Give an account of positive and negative aspects, or advantages and disadvantages of two items.
Compare and contrast	2.2.4	Map, match. Give an account of similarities and differences of two items.
Construct	2.1.3, 2.1.4	Produce, create. *Put elements together to form a coherent whole; reorganise into a new pattern or structure.*
Convert	1.2.2	Transform, express in another way or manner.
Correct	3.2.3	Rectify, make changes for improvement.
Define	3.1.1	Re-define, say in another way, and give more related details or information.
Describe	1.2.2, 1.2.3, 1.3.3, 2.1.1, 2.2.2, 2.2.3, 4.1.1, 4.2.4	Give a detailed account or give more information on the “what” and “how”.
Design	2.1.4, 2.2.3, 4.2.3	Plan, visible thinking, evidence of ideas e.g. an artefact, a computer program, an outcome, a diagram, a drawing or a model.
Explain	1.3.4, 2.2.1, 2.2.5, 4.2.4, 4.2.5	Construct models. Includes elements of analysis. Give a detailed account or give more information on the “where”, “how” and “why”.
Evaluate	2.1.2	*Make judgements based on criteria and standards.*
Identify	2.2.1, 3.1.3, 4.2.1	Recognise, name. Ability to differentiate and discriminate.
Justify	4.1.3, 4.1.4, 4.2.1	Give reasons or evidence to support an action or decision.
Locate	3.2.3	Find.
Modify	3.2.3	Make changes.
Perform	3.2.1	Do, carry out.

Term	Learning outcomes	Definition
Produce	3.2.2, 3.2.4, 4.1.5	Construct, create.
Recognise	2.1.2	Identify.
Represent	1.2.1	Present, express, show as.
Specify	4.2.3	State.
State	3.1.1, 3.1.3, 4.2.5	Say or write what something is about.
Solve	3.1.2	Take action to arrive at an outcome or decision.
Tabulate	1.1.1	Put in the form of a table.
Understand	1.1.3, 1.3.1, 1.3.2, 4.1.1, 4.1.2, 4.1.4, 4.2.4, 4.2.6	*Construct meaning from instructional messages, including oral, written, and graphic communication.*
Use	1.1.2, 1.1.3, 1.1.4, 1.3.2, 4.1.2, 4.1.3, 4.1.4	Practise, implement, create, construct.
Validate	4.2.2	Check accuracy of.

The definitions in italics are based on the Revised Bloom's Taxonomy (Anderson and Krathwohl, 2001)[10] in a hand-out by the Iowa State University, Centre for Excellence in Learning and Teaching:

- Remember – retrieve relevant knowledge from long-term memory
- Understand – construct meaning from instructional messages, including oral, written, and graphic communication
- Apply – carry out or use a procedure in a given situation
- Analyse – break material into constituent parts and determine how parts relate to one another and to an overall structure or purpose
- Evaluate – make judgements based on criteria and standards
- Create – put elements together to form a coherent whole; reorganise into a new pattern or structure

[10] Anderson, L.W. (Ed.), Krathwohl, D.R. (Ed.), Airasian, P.W., Cruikshank, K.A., Maye, R.E., Pintrich, P.R., Raths, J., & Wittrock, M.C. (2001). A t*axonomy for learning, teaching and assessing: A revision of Bloom's Taxonomy of Educational Objectives (Complete edition). New York: Longman.*