Singapore Examinations and Assessment Board

CAMBRIDGE International Education

Singapore–Cambridge Secondary Education Certificate (2027)

G3 Computing (Syllabus K349)

CONTENTS

AIMS

The O-Level 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.

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 five 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 Design, 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.

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 one shift on the day of the examination. Each candidate should have the sole use of a personal computer for the purpose of the examination. The candidates should be able to access spreadsheet and programming software (Python and JupyterLab).

SYLLABUS CONTENT

The syllabus consists of five modules as follows:

Module 1	Computing Fundamentals
Module 2	Algorithms and Programming
Module 3	Spreadsheets
Module 4	Networking
Module 5	Impact of Computing

The learning outcomes are shown in the following pages.

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.

QUICK REFERENCE GUIDE FOR PYTHON

1 Identifiers

When naming variables, functions 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

```
s = input("Prompt for data: ")

print("This is a string")

f = open("input.txt", "r")
line = f.readline()
character = f.read(1)
f.close()

with open("output.txt", "w") as f:
    f.write("Output Line\n")
```

4 Import

```
import <module>
from <module> import <name>
```

5 Data Types

Type	Example	Notes
int	`-3`	integer
float	`3.1415926`	real number
bool	`True`	Boolean
str	`"Hello"`	string (immutable)
list	`[2, 3, 5]`	series of values
dict	`{'key': 'value'}`	key-value pairs

6 Assignment

Statement	Notes
`a = 1`	normal assignment
`b += c`	augmented assignment equivalent to `b = b + c`
`x[y] = z`	assigns `z` to index `y` of list `x` or assigns `z` to key `y` of dictionary `x`
`del a`	deletes variable `a`
`del x[y]`	deletes key `y` and its value from dictionary `x`

7 Arithmetic Operators

Operator	Notes
`+ -`	add, subtract
`* /`	multiply, divide
`%`	remainder or modulo
`**`	exponential or power
`//`	floor division

8 Relational Operators

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

9 Boolean Expressions

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

10 Sequence (List/String) Operations

Operator	Notes
`<seq> + <seq>`	concatenation
`<int> * <seq>`	repetition
`<seq>[index]`	indexing
`<seq>[start:stop]`	slicing
`<seq>[start:stop:step]`	slicing with step
`<value> in <seq>`	membership testing

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 Iteration

while loop

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

for loop

```
for i in range(n):
    <statement(s)>

for record in records:
    <statement(s)>
```

13 Functions

```
# Function definition
def <function name> (<parameters>):
    <function body>
    return <return value>

# Function call
<function name>(<arguments>)
```

14 Built-in Functions

(a) Basic Functions

`abs()`	`chr()`	`float()`	`input()`
`int()`	`len()`	`max()`	`min()`
`open()`	`ord()`	`print()`	`range()`
`round()`	`str()`	`<file>.close()`	`<file>.read()`
`<file>.readline()`	`<file>.write()`	`<str>.endswith()`	`<str>.find()`
`<str>.format()`	`<str>.isalnum()`	`<str>.isalpha()`	`<str>.isdigit()`
`<str>.islower()`	`<str>.isspace()`	`<str>.isupper()`	`<str>.lower()`
`<str>.split()`	`<str>.startswith()`	`<str>.upper()`	

(b) Math Module

`ceil()`	`floor()`	`pow()`	`sqrt()`	`trunc()`

(c) Random Module

`randint()`	`random()`

15 Reserved Words

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

`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`		

FORMULAE

Euclidean Distance

$$d = \sqrt{x^2 + y^2}$$

Boolean Arithmetic

Property	AND	OR	XOR
Annulment	$\mathbf{A} \cdot 0 = 0$	$\mathbf{A} + 1 = 1$	
Identity	$\mathbf{A} \cdot 1 = \mathbf{A}$	$\mathbf{A} + 0 = \mathbf{A}$	$\mathbf{A} \oplus 0 = \mathbf{A}$
Idempotent	$\mathbf{A} \cdot \mathbf{A} = \mathbf{A}$	$\mathbf{A} + \mathbf{A} = \mathbf{A}$	
Self-Inverting			$\mathbf{A} \oplus \mathbf{A} = 0$
Complement	$\mathbf{A} \cdot \overline{\mathbf{A}} = 0$	$\mathbf{A} + \overline{\mathbf{A}} = 1$	$\mathbf{A} \oplus \overline{\mathbf{A}} = 1$
Commutative	$\mathbf{A} \cdot \mathbf{B} = \mathbf{B} \cdot \mathbf{A}$	$\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$	$\mathbf{A} \oplus \mathbf{B} = \mathbf{B} \oplus \mathbf{A}$
Absorption	$\mathbf{A} \cdot (\mathbf{A} + \mathbf{B}) = \mathbf{A}$	$\mathbf{A} + (\mathbf{A} \cdot \mathbf{B}) = \mathbf{A}$	

NOTATION

1 *Boolean Operations*

A · **B**	**A** AND **B**
A + **B**	**A** OR **B**
A ⊕ **B**	**A** XOR **B**
$\overline{\mathbf{A}}$	NOT **A**

2 *Miscellaneous Symbols*

=	is equal to
≠	is not equal to
<	is less than
⩽	is less than or equal to
>	is more than
⩾	is more than or equal to