Singapore Examinations and Assessment Board

CAMBRIDGE
International Education

Singapore–Cambridge General Certificate of Education
Advanced Level Higher 2 (2027)

Computing
(Syllabus 9569)

(First year of examination in 2027)

CONTENTS

	Page
AIMS	**3**
ASSESSMENT OBJECTIVES	**3**
SCHEME OF ASSESSMENT	**4**
SPECIFICATION TABLE	**4**
USE OF CALCULATOR	**4**
CENTRE INFRASTRUCTURE FOR LAB-BASED EXAMINATION	**5**
SYLLABUS CONTENT	**5**
LEARNING OUTCOMES	**6**
REFERENCE GUIDE	**14**

AIMS

The aims of the H2 Computing syllabus are to:

1 Acquire knowledge and understanding of core areas in computing like algorithms, data structures, programming, databases, networks and artificial intelligence;

2 Apply computing knowledge and computational thinking skills to solve real-world problems;

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;

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

ASSESSMENT OBJECTIVES

The examination will assess candidates'

AO1 Knowledge and understanding of computing concepts, algorithms, techniques, tools and related ethics

AO2 Application of knowledge and understanding to analyse real-world problems requiring computing solutions

AO3 Design and develop effective computing solutions; test computing solutions

Candidates will demonstrate computational thinking in a range of real-world problems requiring computing solutions. They will be familiar with and can apply fundamental algorithms and data structures; be able to comment on the social, ethical, legal and economic consequences of computing; understand good design principles and implementation considerations for 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, 3 hours, 100 marks)

This paper tests all six modules of the syllabus through six to eight structured questions of different lengths and marks. This paper carries 60% of the total marks and covers assessment objectives AO1–AO3.

Paper 2 (Lab-based examination, 3 hours, 100 marks)

This paper tests modules 1, 2, 3, 4 and 6 of the syllabus through four structured questions of different lengths and marks taken on a laptop. The questions will test candidates' problem-solving ability through the writing of effective and practical algorithms using HTML, CSS and the Python Programming Language. Candidates are also expected to make use of built-in SQL database engine, SQLite, and a Python web application development framework, Flask, appropriately to solve the problems presented in the examination. This paper carries 40% of the total marks and covers assessment objectives AO2 and AO3.

Candidates will submit soft copies of the required work for marking. The allotted time includes time for saving the required work in the candidates' laptop. The duration, weighting, marks and number of questions are as follows:

Paper	Mode	Duration	Weighting	Marks	Number of Questions
1	Written	3 h	60%	100	6–8
2	Lab-based	3 h	40%	100	4

SPECIFICATION TABLE

Assessment Objectives		Paper 1	Paper 2	Overall
AO1	Knowledge and understanding	~20%	–	20%
AO2	Application	~30%	~10%	40%
AO3	Design and develop effective computing solutions, test computing solutions	~10%	~30%	40%
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 laptop for the purpose of the examination. The Centre must ensure that the following software is installed in all the candidates' laptops for the examination:

- Python 3, along with the following Python modules and their dependencies:
  - Flask
  - JupyterLab
  - Scikit-learn
- DB Browser for SQLite
- Notepad++

The Centre will provide each candidate with a **Reference Guide** (see page 14) on programming syntax for Paper 2.

SYLLABUS CONTENT

The syllabus consists of six modules as follows:

Module 1	Programming Fundamentals
Module 2	Data Structures and Algorithms
Module 3	Data and Information
Module 4	Computer Networks
Module 5	Social, Ethical and Security Impact of Computing
Module 6	Emerging Technologies

LEARNING OUTCOMES

The learning outcomes are as follows:

Module 1: Programming Fundamentals

1.1	**Algorithmic Fundamentals**
Ref.	**Learning Outcome**
1.1.1	Apply the fundamental programming constructs of sequence, selection and iteration to control the flow of program execution.
1.1.2	Understand and devise algorithms using pseudocode[1].
1.1.3	Solve problems by decomposing them into smaller and more manageable parts (modular approach).
1.1.4	Solve problems by solving a small version of the problem and gradually extending the solution to bigger versions of the problem (incremental approach).
1.2	**Programming Constructs**
Ref.	**Learning Outcome**
1.2.1	Know and apply the different data types (integers, floating-point numbers, strings, Booleans).
1.2.2	Use common library functions for input/output, string operations and mathematical operations.
1.2.3	Use abstraction (functions and procedures) to modularise problems into chunks of code for reusability and clarity.
1.2.4	Distinguish between the purpose and use of local and global variables in a program that makes use of functions.
1.2.5	Use Python lists and dictionaries for performing insertion, lookup, update and deletion.
1.3	**Coding Standards**
Ref.	**Learning Outcome**
1.3.1	Use indentation and white space.
1.3.2	Use naming conventions (e.g., meaningful identifier names).
1.3.3	Write comments (name of programmer, date written, program description and inline explanations).

[1] No specific syntax for pseudocode is required and answers should be assessed solely on logical correctness.

1.4	**Recursion**
Ref.	**Learning Outcome**
1.4.1	Know essential features of recursion.
1.4.2	Read and write simple recursive algorithms and programs.
1.4.3	Compare recursion with iteration.
1.5	**Data Validation and Program Testing**
Ref.	**Learning Outcome**
1.5.1	Explain the difference between data validation and data verification.
1.5.2	Understand and apply the following data validation techniques: • existence check (i.e., checking for whether input data is already in the system) • format check • length check • presence check • range check • type check • check digit
1.5.3	Identify, explain and correct syntax, logic and runtime errors.
1.5.4	Design appropriate test cases to cover normal, erroneous and boundary conditions for testing programs.
1.5.5	Trace the steps and list the results of recursive (using recursion tree) and non-recursive programs (using trace tables).
1.5.6	Use appropriate error and exception handling techniques.

Module 2: Data Structures and Algorithms

2.1	**Data Structures**
Ref.	**Learning Outcome**
2.1.1	Illustrate the use of and implement the create, insert, and delete operations for stacks and queues (linear and circular).
2.1.2	Illustrate the use of and implement the create, update (edit, insert, delete) and search operations for linear linked lists. Exclude: doubly-linked lists and circular linked lists.
2.1.3	Illustrate the use of and implement the create, update (edit, insert, delete) and search operations for binary trees (including binary search trees). Exclude: editing and deleting nodes from binary search trees.
2.1.4	Illustrate the use of and implement pre-order, in-order and post-order tree traversals, including the application of in-order tree traversal for binary search trees.
2.1.5	Illustrate the use of and implement breadth-first search and depth-first search for binary trees.
2.1.6	Illustrate the use of and implement hash tables (including handling collisions) with given hash functions.
2.1.7	Store data in and retrieve data from text files.
2.2	**Searching and Sorting Algorithms**
Ref.	**Learning Outcome**
2.2.1	Illustrate the use of and implement sorting algorithms: • Insertion sort • Bubble sort • Quicksort • Merge sort
2.2.2	Illustrate the use of and implement searching algorithms: • Linear search • Binary search
2.2.3	Compare and describe the efficiencies of the sort and search algorithms using Big-O notation for time complexity (worst case). Exclude: space complexity.

2.3	**Object-Oriented Programming**
Ref.	**Learning Outcome**
2.3.1	Define and use classes and objects.
2.3.2	Understand encapsulation and how classes support information hiding and implementation independence, including the use of get and set methods.
2.3.3	Understand inheritance and how it promotes software reuse.
2.3.4	Understand polymorphism and how it enables code generalisation. Exclude: method overloading and multiple inheritance.
2.3.5	Illustrate the relationship between classes (including attributes and methods) using class diagrams.

Module 3: Data and Information

3.1	**Data Representation**
Ref.	**Learning Outcome**
3.1.1	Perform the conversion of positive integers between different number bases: decimal, binary and hexadecimal.
3.1.2	State applications of decimal, binary, hexadecimal number systems in computing context.
3.2	**Character Encoding**
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**
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	Show the relationship between tables by drawing entity-relationship (ER) diagrams.
3.3.6	Understand how NoSQL database management systems address the shortcomings of relational database management systems (SQL).
3.3.7	Explain the applications of SQL and NoSQL.
3.3.8	Write SQL statements and use a programming language to work with SQL databases.

Module 4: Computer Networks

4.1	**Computer Network Fundamentals**
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 the domain name system (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.
4.2	**Web Applications**
Ref.	**Learning Outcome**
4.2.1	Describe the differences between web applications and native applications.
4.2.2	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 can: • accept user input (text and image file uploads) • process the input on the local server • store and retrieve data using an SQL database • display the output (as formatted text/images/table)
4.2.4	Test a web application on a local server.

Module 5: Social, Ethical and Security Impact of Computing

5.1	**Issues and Impact of Computing**
Ref.	**Learning Outcome**
5.1.1	Discuss the social, ethical, legal and economic issues of computing and technology, and their impact on society.
5.1.2	Identify situations where the Computer Misuse Act, Protection from Online Falsehoods and Manipulation Act (POFMA), Protection from Harassment Act and/or Personal Data Protection Act (PDPA) may apply.
5.1.3	Understand the ethics and conduct required of a Computing professional.
5.2	**Information Security**
Ref.	**Learning Outcome**
5.2.1	Describe the concepts of data confidentiality, integrity, and availability.
5.2.2	Discuss methods and best practices to protect confidentiality of data: • Authentication (passwords, multiple-factor authentication) • Access control • Encryption (symmetric, asymmetric) • Firewalls
5.2.3	Discuss methods that can be used to verify the integrity of data and ensure nonrepudiation: • Hashes • Checksums • Digital signatures Exclude: blockchain technology
5.2.4	Describe attacks against data availability: • Denial-of-service (DoS) attacks (distributed and non-distributed) • Malware
5.2.5	Describe measures to mitigate against and recover from loss of availability through: • Redundancy and fault-tolerance • Firewalls • Regular backups • Regular monitoring, alerting, maintenance and systems testing

5.3	**Cybersecurity**
Ref.	**Learning Outcome**
5.3.1	Explain how information in Singapore is protected under the Personal Data Protection Act (PDPA) that governs the collection, use, and disclosure of personal data.
5.3.2	Describe how social engineering attacks (phishing, pretexting, shoulder surfing) can be used to compromise personal information.
5.3.3	Understand how malware (worms, viruses, ransomware, trojans) can compromise computer systems.
5.3.4	Understand how denial-of-service (DoS) attacks work, and how they can affect availability of information and services.
5.3.5	Understand common web application vulnerabilities (SQL injection, cross-site request forgery, file inclusion), how they can be exploited, and the impact from such exploitation.
5.3.6	Suggest solutions (input validation, prepared SQL statements, cross-site request forgery tokens, file permissions) to prevent the exploitation of common web application vulnerabilities.
5.3.7	Understand how encryption, digital signatures, and authentication can enhance the security of network applications.

Module 6: Emerging Technologies

6.1	**Artificial Intelligence**
Ref.	**Learning Outcome**
6.1.1	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.
6.1.2	Give examples of common personal and business tasks that can be performed well by AI: face recognition, voice recognition, image classification, spam filtering, game playing, text generation and image generation.
6.1.3	Define Machine Learning (ML) as a technique used in AI and explain the difference between machine learning and traditional programming.
6.1.4	Know the difference between supervised and unsupervised learning, using k-nearest neighbours and k-means as examples.
6.1.5	Use ML to formulate and solve a problem.
6.1.6	Know the steps required for ML: gathering data, preparing data, choosing a model, training, evaluating, tuning parameters and making predictions.
6.1.7	Implement simple ML programs.

REFERENCE GUIDE

1 Python

1.1 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.

1.2 Comments and Documentation Strings

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

1.3 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")
```

1.4 Import

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

1.5 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
`tuple`	`(2, 3, 5)`	series of values (immutable)

1.6 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

1.7 Arithmetic Operators

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

1.8 Relational Operators

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

1.9 Boolean Expression

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

1.10 Sequence (List/String) Operations

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

1.11 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)>
```

1.12 Iteration

while loop

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

for loop

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

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

1.13 Functions

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

# Function call
<function name>(<value>, <name>=<value>)
```

1.14 Object-Oriented Programming

```
# Class definition
class <class name> (<optional parent class>):

    def __init__(self, <parameters>):
        <constructor body>

    def <method name> (self, <parameters>):
        <method body>
```

1.15 Built-in Functions and Attributes

__file__	<file>.readline()	<list>.clear()	ord()	<str>.isalpha()
__name__	<file>.readlines()	<list>.copy()	print()	<str>.isdigit()
abs()	<file>.write()	<list>.index()	range()	<str>.islower()
bin()	float()	<list>.insert()	round()	<str>.isspace()
<bytes>.decode()	hex()	<list>.pop()	staticmethod()	<str>.isupper()
chr()	input()	<list>.remove()	str()	<str>.lower()
dict()	int()	<list>.reverse()	<str>.encode()	<str>.startswith()
<dict>.clear()	len()	<list>.sort()	<str>.endswith()	<str>.upper()
<dict>.copy()	list()	max()	<str>.format()	
<file>.close()	<list>.append()	min()	<str>.index()	
<file>.read()	<list>.extend()	open()	<str>.isalnum()	

csv module	datetime module		math module
reader()	datetime()	<datetime>.day	ceil()
writer()	datetime.now()	<datetime>.hour	exp()
<writer>.writerow()	datetime.strptime()	<datetime>.minute	floor()
	<datetime>.isoformat()	<datetime>.second	log()
	<datetime>.strftime()	timedelta()	pow()
	<datetime>.year	<timedelta>.days	sqrt()
	<datetime>.month	<timedelta>.seconds	trunc()

os.path module	random module	sqlite3 module	sys module
basename()	random()	connect()	exit()
dirname()	randint()	<connection>.commit()	
isdir()	randrange()	<connection>.close()	
isfile()	shuffle()	<connection>.execute()	
join()		<connection>.rollback()	
		<connection>.row_factory	
		<cursor>.fetchone()	
		<cursor>.fetchall()	
		Row	

1.16 Additional Functions and Attributes

sklearn.neighbors module	sklearn.cluster module	sklearn.model_selection module
KNeighborsClassifier()	KMeans()	cross_validate()
<classifier>.classes_	<estimator>.cluster_centers_	train_test_split()
<classifier>.n_features_in_	<estimator>.labels_	
<classifier>.n_samples_fit_	<estimator>.inertia_	
<classifier>.fit()	<estimator>.n_iter_	
<classifier>.get_params()	<estimator>.n_features_in_	
<classifier>.kneighbors()	<estimator>.fit()	
<classifier>.predict()	<estimator>.fit_predict()	
<classifier>.predict_proba()	<estimator>.get_params()	
<classifier>.score()	<estimator>.predict()	
<classifier>.set_params()	<estimator>.score()	
	<estimator>.set_params()	

numpy module		flask module
`amax()`	`linspace()`	`Flask()`
`amin()`	`log()`	`<flask application>.route()`
`append()`	`mean()`	`<flask application>.run()`
`arange()`	`ones()`	`render_template()`
`array()`	`power()`	`request.files`
`<array>.copy()`	`reshape()`	`request.form`
`ceil()`	`resize()`	`request.method`
`concatenate()`	`round()`	`send_from_directory()`
`cumsum()`	`shape()`	`redirect()`
`delete()`	`sqrt()`	`url_for()`
`dot()`	`std()`	`secure_filename()`
`exp()`	`sum()`	`<uploaded file>.save()`
`empty()`	`transpose()`	
`floor()`	`trunc()`	
`insert()`	`zeros()`	

2 SQL Statements

```
CREATE TABLE table_name(
column1_name COLUMN1_TYPE COLUMN1_CONSTRAINTS,
column2_name COLUMN2_TYPE COLUMN2_CONSTRAINTS,
…
PRIMARY KEY (column1_name, column2_name, …),
FOREIGN KEY (column_name) REFERENCES table_name(column_name)
);
```

```
SELECT column1_name, column2_name, …
FROM table_name
WHERE where_expression
ORDER BY order_expression ASC;
```

```
SELECT column1_name, column2_name, …
FROM table_name
WHERE where_expression
ORDER BY order_expression DESC;
```

```
SELECT table1_name.column1_name, table2_name.column2_name, …
FROM table_name, table2_name
WHERE where_expression;
```

```
SELECT table1_name.column1_name, table2_name.column2_name, …
FROM table1_name
INNER JOIN table2_name ON join_expression;
```

```
SELECT table1_name.column1_name, table2_name.column2_name, …
FROM table1_name
LEFT OUTER JOIN table2_name ON join_expression;
```

```
SELECT
COUNT(*),
MAX(column1_name),
MIN(column2_name),
SUM(column3_name),
…
FROM table_name;
```

```
INSERT INTO table_name(column1_name, column2_name, …)
VALUES (column1_value, column2_value, …);
```

```
UPDATE table_name SET
column1_name = column1_expression,
column2_name = column2_expression,
…
WHERE where_expression;
```

```
DELETE FROM table_name
WHERE where_expression;
```

```
DROP TABLE table_name;
```

3 SQLite Types, Constraints, Functions and Operators

Types	Constraints	Functions	Operators				
NULL	NOT NULL	COUNT()	||	/	<	AND	IS
REAL	PRIMARY KEY	MAX()	+	%	<=	OR	IS NOT
INTEGER	AUTOINCREMENT	MIN()	-	=	>	NOT	
TEXT	UNIQUE	SUM()	*	!=	>=		

4 HTML Elements, Attributes and Character References

The first line of a HTML document must be: `<!doctype html>`

Type	Elements	Attributes
Common		`id, class`
Required	`<html>, <head>, <title>, <body>`	
Metadata	`<link>`	`rel, href`
Structure	`<h1>, <h2>, <h3>, <p>, <div>, <span>, <hr>`	
Text and Media	`<b>, <i>`	
	`<a>`	`href`
	`<img>`	`src, alt`
Table	`<table>, <tr>, <th>, <td>`	
Form	`<form>`	`action, enctype, method`
	`<input>`	`name, type, value`
	`<textarea>`	`name`

Character	&	<	>	"
Reference	`&amp;`	`&lt;`	`&gt;`	`&quot;`

5 Jinja2 Filters

`length`	`safe`

6 CSS Properties

Common	Box Model		Typography

`display`
`background`
`color`
	`height`
	`width`
	`border`
	`border-bottom`
	`border-left`
	`border-right`
	`border-top`
	`margin`
	`margin-bottom`
		`margin-left`
		`margin-right`
		`margin-top`
		`padding`
		`padding-bottom`
		`padding-left`
		`padding-right`
		`padding-top`
			`font-family`
			`font-size`
			`font-style`
			`font-weight`
			`text-align`
			`text-decoration`