Computing

Singapore-Cambridge General Certificate of Education Advanced Level Higher 2

(Syllabus 9569)

(For first year of examination in 2020)

CONTENTS

Singapore Examinations and Assessment Board

CAMBRIDGE International Examinations

AIMS

The syllabus aims to develop problem-solving and computational thinking skills in students, as well as 21CC that will help them to adapt to advancements in the field of computing and global changes impacting the workplace and society. Students would acquire fundamental knowledge in core areas of computing and draw connections to real-world problems by applying these knowledge, skills and attitudes to solve a variety of authentic learning tasks. The main aims of the syllabus are to:

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

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

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

4 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:

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.

Students 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 EXAMINATION

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

Paper 1 (Written examination, 3 hours)

This paper tests all four sections of the syllabus through six to eight structured questions of different lengths and marks. This paper carries 60 per cent of the total marks and covers assessment objective AO1–AO3.

Paper 2 (Lab-based examination, 3 hours)

This paper tests all four sections of the syllabus through four structured questions of different lengths and marks taken in a computer laboratory. The questions will test candidates' problem-solving ability through the writing of effective and practical algorithms using HTML, CSS and the Python Programing 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 per cent 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' computer.

The duration, weighting, marks and number of questions are as follows:

Paper	Mode	Duration	Weighting	Marks	No of Questions
1	Written	3 hours	60%	100	6–8
2	Lab-based	3 hours	40%	100	4

SPECIFICATION GRID

Assessment Objectives	Paper 1	Paper 2	Overall
AO1 Knowledge and understanding	~20%	0%	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.

QUICK REFERENCE GUIDE

Candidates will be provided in the lab-based examination with a quick reference guide of programming syntax.

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. Computers should be installed with the following software for the examination:

- Python 3, along with the following Python modules and their dependencies:
  - Flask
  - PyMongo
  - Jupyter
- DB Browser for SQLite
- MongoDB Community Server
- Notepad++

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.

CONTENT OUTLINE

The syllabus consists of four sections: (1) Algorithms and Data Structures, (2) Programming, (3) Data and Information, and (4) Computer Networks.

Section 1: Algorithms and Data Structures
This section introduces the implementation of data structures to store and retrieve data efficiently, as well as their associated algorithms with the aim of developing problem solving skills. It also includes important software development concepts of decomposition and modularity, as well as techniques such as the use of decision tables to test if algorithms work as intended. Students will need to abstract both data and procedures when they apply computational thinking to a problem. Students will also learn to implement various search and sort algorithms, and compare their efficiencies for evaluation purposes.

Section 2: Programming
This section introduces students to the fundamental principles of programming in scripting languages. Students will learn about the common standards of programming style, programming constructs and library functions to be able to develop their own programs to solve a variety of problems. They are also required to write code to implement data structures such as stacks, queues, linked lists and binary search trees. In addition, students will be expected to design, test and debug their own programs to ensure that they can work through lab-based practical assignments. The fundamental concepts of encapsulation, inheritance and polymorphism associated with object-oriented programming are also covered in this section.

Section 3: Data and Information
This section introduces students to the design, use and application of database management systems. The topics include relational data model, relational query languages and conceptual data design and modelling for relational database design. Students are expected to write programs to retrieve data from either a relational or non-relational database, process the data and return the processed data as a result. The use of databases also highlights the importance of data privacy and integrity. Students should be able to describe measures to safeguard the use of data. In addition, students should be able to describe the code of conduct of a computing professional and discuss the social, economic and ethical implications of computing and technology.

Section 4: Computer Networks
This section provides a broad view of the different types of basic networks, communication protocols and standards in a network. Students will be expected to understand concepts and techniques for developing web applications, describe the different types of threats to network security and propose mechanisms to protect and secure access to networks. They need to design, develop and test web applications as a consolidation of knowledge and skills through hands-on practical work and projects.

LEARNING OUTCOMES

The learning outcomes for each section are as follows:

Section 1: Algorithms and Data Structures

Units

1.1 Algorithmic Representation

1.2 Fundamental Algorithms

1.3 Data Structures

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

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

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

Section 2: Programming

Units

2.1 Coding Standards

2.2 Programming Elements and Constructs

2.3 Implementing Algorithms and Data Structures

2.4 Data Validation and Program Testing

2.5 Fundamentals of Object-Oriented Programming.

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

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

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

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

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

Section 3: Data and Information

Units

3.1 Data Representation

3.2 Character Encoding

3.3 Databases and Data Management

3.4 Social, Ethical, Legal and Economic Issues.

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

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

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

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

Section 4: Computer Networks

Units

4.1 Fundamentals of Computer Networks

4.2 Web Applications

4.3 Network Security

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

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

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