Accreditation Period
2017–2020

Victorian Certificate of Education

ALGORITHMICS

(Higher Education Scored Study)

STUDY DESIGN

www.vcaa.vic.edu.au

VICTORIAN CURRICULUM AND ASSESSMENT AUTHORITY

VICTORIA State Government

Updated January 2018

Authorised and published by the Victorian Curriculum and Assessment Authority
Level 1, 2 Lonsdale Street, Melbourne VIC 3000

Accredited by the Victorian Registration and Qualifications Authority
Level 4, 2 Lonsdale Street, Melbourne VIC 3000

ISBN: 978-1-925264-51-7

The VCE Algorithmics Study Design has been developed and written as a collaborative project by the Victorian Curriculum and Assessment Authority (VCAA), Monash University and the University of Melbourne.

Updated January 2018

Contents

Important information **4**

Introduction **5**
Scope of study 5
Rationale 5
Aims 5
Structure 6
Entry 6
Duration 6
Changes to the study design 6
Monitoring for quality 6
Safety and wellbeing 6
Employability skills 7
Legislative compliance 7

Assessment and reporting **8**
Satisfactory completion 8
Levels of achievement 8
Authentication 8

Unit 3: Algorithmic problem solving **9**
Area of Study 1 9
Area of Study 2 10
Area of Study 3 11
School-based assessment 12
External assessment 13

Unit 4: Principles of algorithmics **14**
Area of Study 1 14
Area of Study 2 15
Area of Study 3 16
School-based assessment 17
External assessment 18

Important information

Accreditation period

Units 3 and 4: 1 January 2017 – 31 December 2020

Implementation of this study commences in January 2017.

Sources of information

The *VCAA Bulletin* is the only official source of changes to regulations and accredited studies. The *VCAA Bulletin* also regularly includes advice on VCE studies. It is the responsibility of each VCE teacher to refer to each issue of the *VCAA Bulletin*. The *VCAA Bulletin* is available as an e-newsletter via free subscription on the VCAA's website at: www.vcaa.vic.edu.au.

The current *VCE and VCAL Administrative Handbook* contains essential information on assessment processes and other procedures.

VCE providers

Throughout this study design the term 'school' is intended to include both schools and other VCE providers.

Copyright

Introduction

Scope of study

The study investigates algorithmics, which provides a structured framework for solving real-world, practical problems with computational methods. Algorithmics is fundamental to computer science and software engineering and is essential for understanding the technical underpinnings of the information society. Beyond its use in computing, algorithmics provides a general discipline of rational thought by virtue of the methodical way it approaches problem solving.

VCE Algorithmics (HESS) examines how information about the world can be systematically represented and how the processes can be made sufficiently explicit and precise so they can be implemented in a computer program. The focus is not on coding but on 'algorithmic thinking'. Algorithmics covers systematic methods for analysing real-world problems and identifying the salient aspects that need to be modelled as the basis for finding a solution. It explores the design of algorithms to solve these problems, resulting in a powerful approach to manipulating, and reasoning about, structured information.

Mathematical techniques are used to establish crucial properties of algorithms, such as how their performance can be scaled to the size of the problem to be solved. This leads to an understanding of what types of algorithms are able to work efficiently at very large scales. Algorithmics also covers deeper topics in computer science such as the possibility of artificial intelligence and the potential for new models of computation inspired by physical and biological systems. This investigation of theoretical topics is complemented by the development of skills in a high-level programming language.

Rationale

Computing is central to our society and economy and drives innovation across many fields of human endeavour. Computation has fundamentally transformed the way we conduct science and engineering; simulation, virtual experiments and computational analysis and prediction have become indispensable parts of the contemporary scientific method. Computation enables us to make sense of data, whether it concerns the environment, the economy, health, entertainment, social and organisational structures or any other sphere of human experience. Algorithmics underpins all computational methods and only through using algorithms can there be full appreciation of their potential and limitations, allowing the development of efficient computational solutions.

VCE Algorithmics (HESS) provides the foundation for studying computer science and software engineering at tertiary level and some universities may offer accelerated pathways to students who have completed this study. The study also provides a conceptual framework for structured problem solving in STEM (Science, Technology, Engineering and Mathematics) and other disciplines that benefit from formal reasoning.

Aims

This study enables students to:

- understand the mathematical foundations of computer science and software engineering
- use symbolic representations and abstraction to formalise real-world information problems
- design algorithms to solve practical information problems, using suitable abstract data types and algorithm design patterns
- investigate the efficiency and correctness of algorithms through formal analysis and empirically through implementation as computer programs
- reason about the physical, mathematical and philosophical limits of computability.

Structure

The study is made up of two units.

Unit 3: Algorithmic problem solving

Unit 4: Principles of algorithmics

Each unit deals with specific content contained in areas of study and is designed to enable students to achieve a set of outcomes for that unit. Each outcome is described in terms of key knowledge and key skills.

Entry

The following list identifies important assumed mathematics knowledge that underpins the study design:

- sets and set operations (union, intersection)
- substitution and transposition in linear and non-linear relations
- the construction of tables of values from a given formula
- development of formulas from word descriptions
- sequences and linear relations generated by recursion
- exponents and logarithms
- the ability to produce and interpret numerical plots.

Most of this assumed knowledge is covered in VCE Mathematics Methods Units 1 and 2. Students are expected to be currently enrolled in, or have successfully completed, VCE Mathematical Methods Units 1 and 2.

Duration

Each unit involves at least 50 hours of scheduled classroom instruction. In addition to this at least 50 hours of self-guided study are expected per unit.

Changes to the study design

During its period of accreditation minor changes to the study will be announced in the *VCAA Bulletin*. The *VCAA Bulletin* is the only source of changes to regulations and accredited studies. It is the responsibility of each VCE teacher to monitor changes and advice about VCE studies published in the *VCAA Bulletin*.

Monitoring for quality

As part of ongoing monitoring and quality assurance, the VCAA will periodically undertake an audit of VCE Algorithmics (HESS) to ensure the study is being taught and assessed as accredited. The details of the audit procedures and requirements are published annually in the *VCE and VCAL Administrative Handbook*. Schools will be notified if they are required to submit material to be audited.

Safety and wellbeing

It is the responsibility of the school to ensure that duty of care is exercised in relation to the health and safety of all students undertaking the study. For this study this means an ergonomically sound work environment.

Employability skills

This study offers a number of opportunities for students to develop employability skills.

Legislative compliance

When collecting and using information, the provisions of privacy and copyright legislation, such as the Victorian *Privacy and Data Protection Act 2014* and *Health Records Act 2001*, and the federal *Privacy Act 1988* and *Copyright Act 1968*, must be met.

Assessment and reporting

Satisfactory completion

The award of satisfactory completion for a unit is based on the teacher's decision that the student has demonstrated achievement of the set of outcomes specified for the unit. Demonstration of achievement of outcomes and satisfactory completion of a unit are determined by evidence gained through the assessment of a range of learning activities and tasks.

Teachers must develop courses that provide appropriate opportunities for students to demonstrate satisfactory achievement of outcomes.

The decision about satisfactory completion of a unit is distinct from the assessment of levels of achievement. Schools will report a student's result for each unit to the VCAA as S (Satisfactory) or N (Not Satisfactory).

Levels of achievement

Units 3 and 4

The VCAA specifies the assessment procedures for students undertaking scored assessment in Units 3 and 4. Designated assessment tasks are provided in the details for each unit in the VCE study designs.

The student's level of achievement in Units 3 and 4 will be determined by two School-assessed Tasks (SATs) as specified in the VCE study design, and external assessment.

The VCAA will report the student's level of achievement on each assessment component as a grade from A+ to E or UG (ungraded). To receive a study score the student must achieve two or more graded assessments and receive S for both Units 3 and 4. The study score is reported on a scale of 0–50; it is a measure of how well the student performed in relation to all others who took the study. Teachers should refer to the current *VCE and VCAL Administrative Handbook* for details on graded assessment and calculation of the study score. Percentage contributions to the study score in VCE Algorithmics (HESS) are as follows:

- Unit 3 School-assessed Task: 20 per cent
- Unit 4 School-assessed Task: 20 per cent
- End-of-year examination: 60 per cent.

Details of the assessment program are described in the sections on Units 3 and 4 in this study design.

Authentication

Work related to the outcomes of each unit will be accepted only if the teacher can attest that, to the best of their knowledge, all unacknowledged work is the student's own. Teachers need to refer to the current *VCE and VCAL Administrative Handbook* for authentication procedures.

Unit 3: Algorithmic problem solving

This unit focuses on how algorithms are used for solving complex problems. Algorithms are systematic problem-solving procedures that exist independently of computers. The study of algorithms lies at the heart of computer science and provides the formal foundation for computer programming. Algorithmic problem solving is a technique that can be applied very broadly in addressing a wide range of complex practical problems.

In Area of Study 1 students develop and apply a range of knowledge and skills to model real-world information problems. This includes the design of data structures for a problem that will be further considered in Area of Study 2. In Area of Study 2 students learn how to design algorithms following a variety of simple algorithm design patterns. They apply this knowledge to design and implement an algorithm that works on the data structures determined in Area of Study 1. The programming requirements for Area of Study 2 will be published annually by the VCAA in the *VCAA Bulletin*. In Area of Study 3 students develop and apply knowledge and skills for evaluating and documenting solutions. All areas of study contribute to the SAT for Unit 3.

In both Units 3 and 4 students are not required to know about the implementation of abstract data types (ADTs), as the main focus of this study is on algorithmic thinking using ADTs rather than on the details of how ADTs are implemented.

Area of Study 1

Data modelling with abstract data types

In this area of study, students develop and apply knowledge and skills in representing information. Students select appropriate abstract data types (ADTs) and use them to model key aspects of real-world problems. ADTs facilitate the modular representation of information and ensure that data is structured and accessed appropriately and that it remains consistent. Students study ADTs with a focus on the graph ADT, which encapsulates a set of nodes along with their interconnections. Students explore how graph ADTs can be used to solve problems involving networks in areas such as social networks, transport networks and the web. For the purpose of this study, the term ADT is limited to: list, array, dictionary (associative array), stack, queue, priority queue and graph.

Outcome 1

On completion of this unit the student should be able to devise formal representations for modelling various kinds of information problems using appropriate abstract data types, and apply these to a real-world problem.

To achieve this outcome the student will draw on key knowledge and key skills outlined in Area of Study 1.

Key knowledge

- the motivation for using ADTs
- modularisation and abstraction of information representation with ADTs
- signature specifications of ADTs using operator names, argument types and result types
- specification and uses of the following ADTs:
  - list, array, dictionary (associative array)
  - stack, queue, priority queue
- specification and uses of the graph ADT, including:
  - cyclicity and connectedness as properties of graphs
  - trees and decision trees
  - directed graphs and directed acyclic graphs
  - weighted graphs, weighted path lengths and topological distance.

Key skills

- explain the role of ADTs for data modelling
- read and write ADT signature specifications
- use ADTs in accordance with their specifications
- select and apply appropriate ADTs to model real-world problems
- model basic network and planning problems with graphs.

Area of Study 2

Algorithm design

In this area of study, students learn how to formalise processes as algorithms and to execute them automatically. They use the language of algorithms to describe general approaches to problem solving and to give precise descriptions of how specific problems can be solved. Students learn how to decompose problems into smaller parts that can be solved independently. This forms the basis of modularisation. Students explore a variety of problem-solving strategies and algorithm design patterns. Students explore example applications of these design patterns and learn about their implications for efficient problem solutions. They learn about recursion as a method for constructing solutions to problems by drawing on solutions to smaller instances of the same problem. The programming language used to implement the algorithms as a computer program must explicitly support the ADTs listed in the key knowledge in Area of Study 1 either directly or by using a library.

Outcome 2

On completion of this unit the student should be able to design algorithms to solve information problems using basic algorithm design patterns, and implement the algorithms.

To achieve this outcome the student will draw on key knowledge and key skills outlined in Area of Study 2.

Key knowledge

- basic structure of algorithms
- pseudocode concepts, including variables and assignment, sequence, iteration, conditionals and functions
- programming language constructs that directly correspond to pseudocode concepts
- conditional expressions using the logical operations of AND, OR, NOT
- recursion and iteration and their uses in algorithm design
- modular design of algorithms and ADTs
- characteristics of algorithm design patterns, including brute-force search, greedy methods, decrease and conquer
- graph traversal techniques: breadth-first search, depth-first search and best-first search
- specification, correctness and limitations of the following graph algorithms:
  - Prim's algorithm for computing the minimal spanning tree of a graph
  - Dijkstra's algorithm and the Bellman-Ford algorithm for the single-source shortest path problem
  - the Floyd-Warshall algorithm for transitive closure
  - Floyd's algorithm for the all-pair shortest path problem
  - the PageRank algorithm for estimating the importance of a node based on its links.

Key skills

- interpret pseudocode and execute it manually on given input
- write pseudocode
- identify and correct errors in pseudocode
- apply algorithm design patterns
- select appropriate graph algorithms and justify the choice based on their properties and limitations
- demonstrate the correctness of the specified graph algorithms
- read and design appropriate recursive and iterative algorithms
- write modular algorithms using ADTs and functional abstractions
- describe how complex information can be represented by a combination of ADTs
- use search methods on decision trees and graphs to solve planning problems
- implement algorithms including graph algorithms as computer programs in a very high-level programming language that directly supports a graph ADT.

Area of Study 3

Applied algorithms

In this area of study, students combine their knowledge of data modelling and algorithm design to solve real-world problems. Students consider a variety of algorithms and ADTs before selecting a suitable combination. They evaluate their chosen combination of algorithms and data types relative to other possible choices, justify their decisions and document the results of the evaluation. Students consider if the combination of algorithms and data models are fit for purpose. Typically this could be measured in terms of the selection of salient features to achieve an appropriate level of abstraction, the modular representation of data using ADTs and the quality of result produced by the algorithm.

In this area of study students develop and apply practical skills on the basis of theoretical knowledge developed in Areas of Study 1 and 2, to evaluate and document a data model and algorithm developed in these areas of study.

Outcome 3

On completion of this unit the student should be able to evaluate and document algorithms and data representations, and solve a real-world problem, the solution for which requires the integration of algorithms and data types.

To achieve this outcome the student will draw on key knowledge and key skills outlined in Area of Study 3.

Key knowledge

- characteristics and applicability of ADTs and algorithm design patterns
- suitability of ADTs and algorithm design patterns for a variety of problem contexts
- combinations of ADTs to meet complex problem requirements
- induction and contradiction as methods for demonstrating the correctness of brute force, greedy methods and decrease and conquer algorithms.

Key skills

- justify suitable ADTs and algorithm design patterns
- evaluate the adequacy of a given combination of algorithms and ADTs for solving problems, relative to other possible choices
- propose an argument for the correctness of an algorithm
- communicate the design of data models and algorithms
- communicate the meaning of computed solutions in terms of the original real-world problem to be solved.

School-based assessment

Satisfactory completion

The award of satisfactory completion for a unit is based on whether the student has demonstrated the set of outcomes specified for the unit. Teachers should use a variety of learning activities and assessment tasks to provide a range of opportunities for students to demonstrate the key knowledge and key skills in the outcomes.

The areas of study and key knowledge and key skills listed for the outcomes should be used for course design and the development of learning activities and assessment tasks.

Assessment of levels of achievement

School-assessed Task

The student's level of achievement in Unit 3 will be assessed through a School-assessed Task that includes Outcomes 1, 2 and 3. Details of the School-assessed Task for Unit 3 are set out in the following table.

Contribution to final assessment

The Unit 3 School-assessed Task will contribute 20 per cent to the study score.

Outcomes	Assessment tasks
Outcome 1 Devise formal representations for modelling various kinds of information problems using appropriate abstract data types, and apply these to a real-world problem. **Outcome 2** Design algorithms to solve information problems using basic algorithm design patterns, and implement the algorithms. **Outcome 3** Evaluate and document algorithms and data representations, and solve a real-world problem, the solution for which requires the integration of algorithms and data types.	A folio, including: • two to four tasks using a range of abstract data types to model the salient aspects of problems • two to four tasks using a range of algorithm design patterns to specify algorithms to solve problems. **AND** A written explanation of each of: • the specification and application of ADTs • the specification and application of algorithms for graphs. (approximately 45–60 minutes for each explanation) **AND** A project consisting of three connected components: • A data model of a real-world problem, including: – specification of the data model – a concrete instance of the data model (worked example), and – documentation of the data model development approach. (approximately 300–500 words) • An algorithm to solve a real-world problem that builds on an existing data model including: – pseudocode to solve the problem – implementation of the algorithms in a high-level programming language making appropriate use of the standard ADTs, and – documentation of the algorithm development approach. (approximately 300–500 words) • An evaluation of an existing data model and algorithm in the form of a written report. (approximately 300–500 words)

External assessment

The level of achievement for Units 3 and 4 is also assessed by an end-of-year examination, which will contribute 60 per cent.

Unit 4: Principles of algorithmics

This unit focuses on the performance of algorithms and the scope and limitations of algorithms. Students develop the knowledge and skills to identify the resources that an algorithm needs to function efficiently and effectively. In Area of Study 1 students investigate the efficiency of algorithms and apply this to the formal analysis of a naïve algorithm for a given problem. They also learn about soft limits of computability, namely, problems that can in principle be solved, but cannot be solved for practical problem sizes due to time or space constraints. In Area of Study 2 students learn about a variety of more sophisticated algorithm design patterns, and apply their knowledge of these to construct an improved solution for the problem posed in Area of Study 1. In Area of Study 3, students learn about hard limits of computability, namely, problems the solution of which cannot be computed at all by any kind of computational machinery. A list of alternative methods of computation will be published annually by the VCAA in the *VCAA Bulletin*.

Area of Study 1

Formal algorithm analysis

In this area of study, students investigate the efficiency of algorithms using mathematical techniques. Students learn how some computable problems require such a large amount of resources that in practice it is not possible to solve these exactly for realistic problem sizes. Students examine specific, widely occurring instances of such problems and the reasons why these problems cannot be solved. Students analyse time complexity formally and informally, while they study space complexity as a general concept. The naïve algorithm analysed in this area of study will form the basis for more sophisticated development in Area of Study 2.

Outcome 1

On completion of this unit the student should be able to establish the efficiency of simple algorithms and explain soft limits of computability.

To achieve this outcome the student will draw on key knowledge and key skills outlined in Area of Study 1.

Key knowledge

- the concept of algorithm complexity including time and space complexity
- the concept of P and NP-complete complexity classes
- the concepts of Big-O, Big-Ω and Big-Θ notation
- differences between best case and worst case complexity analysis of algorithms
- the P time complexity class and examples of algorithms that have complexities of O(1), O(log *n*), O(*n*), O(*n* log *n*), O(n^2) and O(n^3)
- an understanding that problems can be harder than P problems and the implications for their solvability
- consequences of combinatorial explosions and indicators for them
- recurrence relations as a method of describing the time complexity of recursive algorithms
- the Master Theorem for solving recurrence relations of the form:

$$T(n) = \begin{cases} aT(\frac{n}{b}) + kn^c & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases}$$

where $a > 0, b > 1, c \geq 0, d \geq 0, k > 0$

and its solution: $T(n) = \begin{cases} O(n^c) & \text{if } \log_b a < c \\ O(n^c \log n) & \text{if } \log_b a = c \\ O(n^{\log_b a}) & \text{if } \log_b a > c \end{cases}$

Key skills

- formally analyse the efficiency of algorithms using Big-O notation
- compare algorithms based on their algorithmic complexity
- identify input patterns that lead to best and worst case performance
- read off a recurrence relation for the running time of a recursive algorithm that can be solved by the Master Theorem or takes the form: $T(n) = \sum_{i=1}^{k} T(n - a_i) + b$, where $a_i \in N$
- use the stated Master Theorem to solve recurrence relations
- demonstrate how exponentially sized search and solution spaces impose practical limits on computability.

Area of Study 2

Advanced algorithm design

In this area of study, students examine more advanced algorithm design patterns. Students learn how to select algorithmic approaches from a wider range of options, depending on the structure of the problem that is being addressed. They investigate how some problems are solvable in principle while being intractable in practice. They explore examples of such problems with real-world relevance and learn how such problems can be tackled by computing near-optimal solutions. Students apply their knowledge and skills to improve the naïve algorithm analysed in Area of Study 1.

Outcome 2

On completion of this unit the student should be able to solve a variety of information problems using algorithm design patterns and explain how heuristics can address the intractability of problems.

To achieve this outcome the student will draw on key knowledge and key skills outlined in Area of Study 2.

Key knowledge

- divide and conquer algorithms that have linear time split and merge steps, including mergesort and quicksort
- dynamic programming algorithms that require no more than a single dimension array for storage, including Fibonacci numbers, 1-D knapsack problem, change making problem
- tree search by backtracking and its applications
- differences between divide and conquer and dynamic programming and their applications
- induction and contradiction as methods for demonstrating the correctness of dynamic programming and backtracking algorithms
- minimax method to solve combinatorial problems
- heuristics and randomised meta-heuristic algorithms, including simulated annealing, as approaches to overcome soft limits of computation
- limitations of heuristics and randomised meta-heuristic algorithms
- the graph colouring, the 1-D knapsack and travelling salesman problems and heuristic methods for solving them.

Key skills

- recognise and apply the divide and conquer, dynamic programming, and backtracking design patterns
- apply the minimax method
- develop and compare different algorithms for solving the same problem, using different algorithm design patterns
- propose an argument for the correctness of an algorithm

- use heuristics to solve computationally hard problems
- explain the meaning of randomised approaches for intractable problems, including the graph colouring, the 1-D knapsack and travelling salesman problems.

Area of Study 3

Universality of computation and algorithms

In this area of study, students examine computation as a universal concept that is independent of programming languages and computer hardware. They explore examples of computational problems that cannot be solved in principle and learn formal methods for studying the outer limits of computation. They also study alternative methods of computation. Students investigate the prospects for automatic reasoning and engage with the philosophical debate about whether artificial intelligence is possible. They study these topics in historical context to explore past and present connections between computer science, mathematics and philosophy. Students are not required to produce proofs or formal explanations concerning undecidability.

Outcome 3

On completion of this unit the student should be able to explain the scope of algorithmics as an approach to computational problem solving and the universality of computation, and its limits, using core concepts from theoretical computer science.

To achieve this outcome the student will draw on key knowledge and key skills outlined in Area of Study 3.

Key knowledge

- David Hilbert's 1927 program to fully formalise mathematical reasoning, including its goals, historical context, outcome and connection to the origin of computer science
- characteristics of a Turing machine
- the concept of decidability and undecidability, including the Halting Problem and the demonstration that it is undecidable, and implications for automatic program verification
- the meaning of the Church-Turing Thesis and its limitations
- implications of undecidability for Hilbert's Program
- the relationship between Turing machines and computational complexity theory
- the meaning of Cobham's thesis for computational complexity theory
- John Searle's Chinese Room Argument, including:
  - historical context and motivation
  - connections to artificial intellligence
  - standard responses to the Chinese Room argument, both for and against
- characteristics of alternative methods of computation.

Key skills

- demonstrate the existence of hard limits of computability
- discuss the merit and limitations of the Church-Turing and Cobham theses
- define decidability and undecidability and explain selected examples
- explain conceptually how the equivalence of computational formalisms is shown
- present and evaluate responses to the Chinese Room Argument
- explain how alternative methods of computation might be used to overcome current limits of computation.

School-based assessment

Satisfactory completion

The award of satisfactory completion for a unit is based on whether the student has demonstrated the set of outcomes specified for the unit. Teachers should use a variety of assessment tasks to provide a range of opportunities for students to demonstrate the key knowledge and key skills in the outcomes.

The areas of study and key knowledge and key skills listed for the outcomes should be used for course design and the development of learning activities and assessment tasks.

Assessment of levels of achievement

School-assessed Task

The student's level of achievement in Outcomes 1, 2 and 3 in Unit 4 will be assessed through a School-assessed Task. Details of the School-assessed Task for Unit 4 are set out in the following table.

Contribution to final assessment

The School-assessed Task will contribute 20 per cent to the study score.

Outcomes	Assessment tasks
Outcome 1 Establish the efficiency of simple algorithms and explain soft limits of computability.	A written explanation of: • formal analysis techniques and the practical limits of computability • algorithm design patterns and techniques for addressing the limits of computation. (approximately 45–60 minutes)
	AND
Outcome 2 Solve a variety of information problems using algorithm design patterns and explain how heuristics can address the intractability of problems.	The design of an algorithm, consisting of two components: • formal analysis of a given naïve algorithm (approximately 400 words) • a response to a naïve algorithm consisting of: – an improved algorithm design – an analysis of the improved design, including its correctness. (approximately 600 words)
	AND
Outcome 3 Explain the scope of algorithmics as an approach to computational problem solving and the universality of computation, and its limits, using core concepts from theoretical computer science.	An explanation of the universality of computation and algorithms in one or more of the following forms: • a written report (approximately 700–800 words) • visual report (images supported by approximately 400–500 words) • an oral report (10–15 minutes).

External assessment

The level of achievement for Units 3 and 4 is also assessed by an end-of-year examination.

Contribution to final assessment

The examination will contribute 60 per cent.

End-of-year examination

Description

The examination will be set by a panel appointed by the VCAA. All the key knowledge and key skills that underpin the outcomes in Units 3 and 4 are examinable.

Conditions

The examination will be completed under the following conditions:

- Duration: two hours.
- Date: end-of-year, on a date to be published annually by the VCAA.
- VCAA examination rules will apply. Details of these rules are published annually in the *VCE and VCAL Administrative Handbook.*
- The examination will be marked by assessors appointed by the VCAA.

Further advice

The VCAA publishes specifications for all VCE examinations on the VCAA website. Examination specifications include details about the sections of the examination, their weighting, the question format/s and any other essential information. The specifications are published in the first year of implementation of the revised Units 3 and 4 sequence together with any sample material.