VCE Algorithmics (HESS) Administrative information for School-based Assessment in 2017

Unit 4 School-assessed Task

The School-assessed Task for Unit 4 contributes 20 per cent to the study score.

Teachers will provide to the Victorian Curriculum and Assessment Authority (VCAA) with a score against each criterion that represents an assessment of the student's level of performance for Unit 4 Outcome 1, Outcome 2 and Outcome 3. The recorded scores must be based on the teacher's assessment of the student's performance according to the criteria on pages 5–15. This assessment is subject to the VCAA's statistical moderation process.

The 2017 VCE Algorithmics (HESS) assessment sheet on page 18 is to be used by teachers to record the Unit 4 School-assessed Task score. The completed assessment sheet for each student's School-assessed Task must be available on request by the VCAA. Student scores for the Unit 4 School-assessed Task must be submitted via VASS no later than 6 November 2017.

The performance descriptors for the assessment criteria are published annually on the Algorithmics (HESS) study page of the VCAA website. Details of authentication requirements and administrative arrangements for School-assessed Tasks are published in the *VCE and VCAL Administrative Handbook 2017*.

The School-assessed Task for Unit 4 relates to:

- Outcome 1
- Outcome 2
- Outcome 3

Unit 4

Formal algorithm analysis

Outcome 1

Establish the efficiency of simple algorithms and explain soft limits of computability.

Nature of tasks

- A written explanantion of formal analysis techniques and the practical limits of computability (approximately 45–60 minutes).
- Formal analysis of a given naïve algorithm (approximately 400 words).

Scope of tasks

Written explanation

The task should enable students to explain in writing analysis techniques and the practical limits of computability. For authentication purposes this task may be completed under test conditions within a timeframe of 45 to 60 minutes in class time. Student performance for this task is assessed through Criterion 1. The criterion should be used to develop a task that allows all students to demonstrate their level of performance.

Algorithm design

Students analyse the efficiency of a naïve algorithm using mathematical techniques. The naïve algorithm should have scope for improvement through the application of one of the algorithm design patterns studied in Outcome 2. It should take a straightforward approach to the problem and not be unnecessarily complicated. The word range for this report is approximately 400 words. Student performance for this task is assessed through Criterion 2. The naïve algorithm analysed by students in this criterion will form the basis of an improved design in Outcome 2, Criterion 4.

Advanced algorithm design

Outcome 2

Solve a variety of information problems using algorithm design patterns and explain how heuristics can address the intractability of problems.

Nature of tasks

- A written explanation of algorithm design patterms and techniques for addressing the limits of computation (approximately 45-50 minutes).
- A response to a naïve algorithm consisting of:
  - an improved algorithm design
  - an analysis of the improved design, including its correctness (approximately 600 words).

Scope of tasks

Written explanation

The task should enable students to explain in writing algorithm design patterns and techniques. For authentication purposes this task may be completed under test conditions within a timeframe of 45 to 60 minutes in class time. Student performance for this task is assessed through Criterion 3. The criterion should be used to develop a task that allows all students to demonstrate their level of performance.

Algorithm design

Students design an improved algorithm in response to the naïve algorithm given in Outcome 1. They analyse the efficiency of the improved algorithm and propose a valid argument for its correctness. The word range for this report is approximately 600 words. Student performance on this task will be assessed through Criterion 4.

Universality of computation and algorithms

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.

Nature of task

- An explanation of the universality of computation and algorithms in one or more of the following forms:
  - a written report (approximately 700–800 words)
  - a visual report (images supported by approximately 400–500 words)
  - an oral report (10–15 minutes).

Scope of task

Explanation

Students develop a report using core concepts from theoretical computer science studied in class. They should be able to give detailed descriptions, explanations and evaluations of these core concepts. The word range for this report is 700 to 800 words. Student performance on this task will be assessed through Criteria 5 and 6.

The following rubric is used to assess student achievement on Unit 4 Outcome 1, Outcome 2 and Outcome 3. Teachers assess evidence produced from the tasks against the criteria and performance descriptors.

The criteria identify specific characteristics that are used to judge levels of performance against the outcomes. Performance descriptors describe typical evidence associated with five different levels of performance for a criterion (five levels; 10 marks).

Note: This rubric is based on the premise that each column includes all evidence of the columns to the left for each criterion, that is, the conditions are cumulative.

The following table shows the relationships between the outcomes, tasks and criteria.

Outcome	Task	Criteria
Outcome 1	A written explanation of formal analysis techniques and the practical limits of computability.	1
Outcome 1	Formal analysis of a given naïve algorithm.	2
Outcome 2	A written explanation of algorithm design patterns and techniques for addressing the limits of computation.	3
Outcome 2	A response to a naïve algorithm consisting of: • an improved algorithm design • an analysis of the improved design, including its correctness.	4
Outcome 3	An explanation of the universality of computation and algorithms in one or more of the following forms: • a written report • a visual report • an oral report.	5 and 6

VCE Algorithmics (HESS): Unit 4 School-assessed Task 2017						
Assessment criteria	**Levels of Performance**					
	Not shown	**1–2 (low)**	**3–4**	**5–6 (medium)**	**7–8**	**9–10 (high)**
1. Understanding of formal algorithm analysis		Describes the meaning of algorithm complexity using everyday language. Uses the written notation of Big-O notation. Orders two different Big-O expressions by their order of growth. States an example of an algorithm with a given time complexity. Informally describes the meaning of intractability and states an example of an intractable problem. Describes the concept of exponential growth using everyday language. Uses the written notation of recurrence relations. States the solutions to some input cases of the Master Theorem.	Describes the meaning of algorithm complexity and describes a motivation for the study of the asymptotic time complexity of algorithms. Describes the difference between time and space complexity using everyday language. Identifies some relevant distinctions between algorithm complexity and the complexity class of problems. Describes the meaning of a given Big-O expression. Identifies that running time may vary between inputs of equal size. Describes some elements of the algorithmic structures that give rise to $O(n^2)$ or $O(n^3)$ time complexities. Describes the meaning of intractability and describe its practical relevance.	Explains the concept of order of growth as it relates to the asymptotic complexity of algorithms. Explains the distinction between the time and space complexity of an algorithm. Explains the concept of the P complexity class. Determines the order of growth associated with a range of mathematical expressions. Determines whether the running time of a specific algorithm varies for different inputs of equal size. Explains why some computable problems are intractable, with the use of examples of both tractable and intractable problems. Explains one indicator of exponentially sized search spaces based on examples introduced in the study.	Explains clearly and precisely the concept of the asymptotic complexity of algorithms. Explains clearly and precisely the distinction between the time and space complexity of an algorithm. Defines clearly and precisely the P and NP-Complete complexity classes. Explains the general mathematical meaning of statements made using Big-O, Big-Ω and Big-Θ notation. Discusses some differences between best case and worst case analysis of algorithms. Explains some elements of the algorithmic structures that give rise to $O(n^2)$, $O(n^3)$,$O(\log(n))$ and $O(n \log(n))$ time complexities.	Compares the practical limits placed on the use of an algorithm by its time complexity with those placed by its space complexity. Compares the P and NP-Complete computational complexity classes through the use of well-chosen example problems. Discusses the practical consequences of a problem belonging to either the P or NP-Complete complexity class. Explains precisely the mathematical meaning of statements made using Big-O, Big-Ω and Big-Θ notation. Discusses the differences between and appropriateness of best and worst case complexity analysis for evaluating algorithms in practice. Explains the algorithmic structures that give rise to $O(n^2)$, $O(n^3)$,$O(\log(n))$ and $O(n \log(n))$ time complexities.

VCE Algorithmics (HESS): Unit 4 School-assessed Task 2017						
Assessment criteria	**Levels of Performance**					
	Not shown	**1–2 (low)**	**3–4**	**5–6 (medium)**	**7–8**	**9–10 (high)**
			Describes the concept of exponential growth in a search space. Evaluates a recurrence relation for a given input. Describes the Master Theorem, including the problem that it solves and the solution it provides, with minor errors or omissions. Makes simple comparisons between the efficiency of algorithms based on their algorithmic complexity.	Explains the Master Theorem, including the problem that it solves and the solution it provides. Makes practical comparisons between the efficiency of algorithms based on their algorithmic complexity, distinguishing between differences in algorithmic complexity that lead to significant changes in the size of tractable inputs and those that do not. Demonstrates some features of how combinatorial algorithms create exponentially sized search spaces.	Demonstrates clearly how combinatorial algorithms create exponentially sized search spaces.	Explains the connections between the call trees produced by recursive functions and the solutions to the three cases of the Master Theorem.

VCE Algorithmics (HESS): Unit 4 School-assessed Task 2017						
Assessment criteria	**Levels of Performance**					
	Not shown	**1–2 (low)**	**3–4**	**5–6 (medium)**	**7–8**	**9–10 (high)**
2. Skills in establishing the efficiency of a naïve algorithm		Identifies lines of the naïve algorithm's pseudocode that execute in constant running time and lines whose running time vary depending on the algorithm's input. Applies step-counting concepts with limited accuracy to reason about the running time of parts of the naïve algorithm. Applies recurrence relation methods with very limited accuracy to a recursive component of the naïve algorithm to analyse its worst case time complexity.	Applies step-counting methods with some accuracy to iterative or conditional components of the naïve algorithm whose running time varies depending on the input to analyse their worst case time complexity. Applies recurrence relation methods with some accuracy to a recursive component of the naïve algorithm to analyse its worst case time complexity. Identifies reasonable estimations for the performance of ADT operations used within the naïve algorithm.	Analyses the worst case time complexity of the naïve algorithm through the selection and application of appropriate techniques. Some errors or omissions in the application of the techniques lead to an overall inaccurate analysis. Accurately applies step-counting methods to a nested iterative component of the naïve algorithm to analyse its worst case time complexity. Accurately applies recurrence relation methods to a recursive component of the naïve algorithm to analyse its worst case time complexity. Identifies a single input instance for the naïve algorithm that would result in the best-case or worst-case running time, respectively.	Analyses clearly and thoroughly the time complexity of the naïve algorithm through the selection and application of appropriate techniques. Describes the class of input instances for the naïve algorithm that would result in the best case or worst case running times, respectively.	Analyses precisely and elegantly the time complexity of the naïve algorithm through the efficient selection and application of appropriate techniques. Describes efficiently and precisely the classes of input instances for the naïve algorithm that would result in the best case and the worst case running times.

VCE Algorithmics (HESS): Unit 4 School-assessed Task 2017						
Assessment criteria	**Levels of Performance**					
	Not shown	**1–2 (low)**	**3–4**	**5–6 (medium)**	**7–8**	**9–10 (high)**
3. Understanding of advanced algorithm design		States an example of a backtracking, divide and conquer or dynamic programming algorithm. Identifies the divide and merge steps within the pseudocode of a given divide and conquer algorithm. States the problems solved by some of the mergesort, quicksort, knapsack, change making and minimax algorithms. States the algorithmic design paradigm associated with the mergesort, quicksort, knapsack or change making algorithms. Describes using everyday language the use of heuristics or randomisation as general approaches to intractable problems.	Identifies the attributes of a given algorithm that indicate the use of a specific algorithmic design pattern. Determines whether a given algorithm is an example of specific algorithmic design paradigm. Outlines the mergesort, quicksort, knapsack, change making or minimax algorithm, possibly with some minor errors. Completes the missing lines of mergesort, quicksort, knapsack, change making and minimax algorithm. Proposes an argument for the correctness of one of the specified divide and conquer or dynamic programming algorithms that does not consider the general case of the problem, rather only the correctness of a specific problem instance.	Describes the general structure of the backtracking, divide and conquer, or dynamic programming algorithmic design patterns. Describes in pseudocode any of the mergesort, quicksort, knapsack, change making or minimax algorithms. Proposes an argument for the correctness of one of the specified divide and conquer or dynamic programming algorithms that considers the general case of the problem but only includes a minority of the required steps in the chain of argument. Outlines a specific algorithm based on either a heuristic or randomisation design approach.	Explains, using appropriate metalanguage, some of the principles of the backtracking, divide and conquer, or dynamic programming algorithmic design patterns. Analyses the benefits and costs associated with any of the mergesort, quicksort, knapsack or change making algorithms in comparison with naive algorithms for the same problem. Proposes an argument for the correctness of one of the specified divide and conquer or dynamic programming algorithms that considers the general case of the problem but not all steps in the chain of argument are included. Describes in detail a specific algorithm based on either a heuristic or randomisation design approach.	Explains precisely, using appropriate metalanguage, the principles of the backtracking, divide and conquer, or dynamic programming algorithmic design patterns. Proposes a valid argument for the correctness of one of the specified divide and conquer or dynamic programming algorithms using either the induction or contradiction method. Discusses the merits and limitations of the use of heuristic and randomised algorithms in pracitce. Compares the decision and optimisation versions of the graph colouring, knapsack or travelling salesman problems with appropriate references to the computational complexity classes of the problems.

VCE Algorithmics (HESS): Unit 4 School-assessed Task 2017						
Assessment criteria	**Levels of Performance**					
	Not shown	**1–2 (low)**	**3–4**	**5–6 (medium)**	**7–8**	**9–10 (high)**
		Describes using everyday language the graph colouring, knapsack or travelling salesman problem. Identifies a decision within the graph colouring, knapsack or travelling salesman problem that could be randomised.	Describes the principles of heuristic or randomisation algorithms as general approaches to intractable problems, with the inclusion of some technical detail and use of appropriate metalanguage. Determines whether a problem given in context is an instance of the graph colouring, knapsack or travelling salesman problem. Describes how a decision within the graph colouring, knapsack or travelling salesman problem could be randomised.	Defines clearly and precisely the graph colouring, knapsack or travelling salesman problem using appropriate metalanguage. Partially describes an algorithm for the graph colouring, knapsack or travelling salesman problem that would be appropriate for large input instances.	Describes some of the limitations of heuristic and randomised algorithms. Explains the decision and optimisation versions of the graph colouring, knapsack or travelling salesman problems. Describes clearly a specific algorithm for the graph colouring, knapsack or travelling salesman problem that would be appropriate for large input instances.	

VCE Algorithmics (HESS): Unit 4 School-assessed Task 2017						
Assessment criteria	**Levels of Performance**					
	Not shown	**1–2 (low)**	**3–4**	**5–6 (medium)**	**7–8**	**9–10 (high)**
4. Skills in developing an improved algorithm in response to a naïve algorithm		Selects an algorithm design pattern and applies limited elements of its structure to design an algorithm that is either incomplete or incorrect. Compares very limited aspects of the student-designed algorithm and the naïve algorithm for the problem. Identifies lines of the improved algorithm's pseudocode that execute in constant running time and lines whose running time vary depending on the algorithm's input. Applies step-counting concepts with limited accuracy to reason about the running time of parts of the improved algorithm. Applies recurrence relation methods with very limited accuracy to a recursive component of the improved algorithm to analyse its worst case time complexity.	Selects an algorithm design pattern and applies significant elements of its structure to develop an algorithm that is not correct. Compares simply the student-designed algorithm and the naïve algorithm for the problem. Proposes an argument for the correctness of the algorithm that does not consider the general case, rather only the correctness of a specific problem instance. Applies step-counting methods with some accuracy to iterative or conditional components of the improved algorithm whose running time varies depending on the input to analyse their worst case time complexity. Applies recurrence relation methods with some accuracy to a recursive component of the improved algorithm to analyse its worst case time complexity.	Selects an appropriate algorithm design pattern and applies it to develop an algorithm that is correct for most inputs and is an improvement in algorithmic complexity on a naive algorithm. Compares in detail the student-designed algorithm and the naïve algorithm for the problem, highlighting the improvements made. Proposes an argument for the correctness of the algorithm that considers the general case but only includes a minority of the required steps in the chain of argument. Analyses the worst case time complexity of the improved algorithm through the selection and application of generally appropriate techniques. Some errors or omissions in the application of the techniques lead to an overall inaccurate analysis.	Selects an appropriate algorithm design pattern and applies it to develop a clearly expressed and correct algorithm that is an improvement in algorithmic complexity on the naive algorithm. Compares comprehensively and accurately the student-designed algorithm and the naïve algorithm for the problem, highlighting the improvements made and any trade-offs required. Proposes an argument for the correctness of the algorithm that considers the general case of the problem but not all steps in the chain of argument are explained. Clearly and thoroughly analyses the time complexity of the improved algorithm through the selection and application of appropriate techniques.	Selects an appropriate algorithm design pattern and applies it to develop an elegantly and precisely expressed, correct and efficient algorithm. Proposes a valid argument for the correctness the algorithm, if appropriate using either the induction or contradiction method. Analyses precisely and elegantly the time complexity of the improved algorithm through the efficient selection and application of appropriate techniques. If relevant, describes efficiently and precisely the class of input instances for the improved algorithm that would result in the best case or the worst case running times.

VCE Algorithmics (HESS): Unit 4 School-assessed Task 2017						
Assessment criteria	**Levels of Performance**					
	Not shown	**1–2 (low)**	**3–4**	**5–6 (medium)**	**7–8**	**9–10 (high)**
			Identifies reasonable estimations for the performance of ADT operations used within the improved algorithm.	Accurately applies recurrence relation methods to a recursive component of the improved algorithm to analyse its worst case time complexity. Accurately applies step-counting methods to a nested iterative component of the improved algorithm to analyse its worst case time complexity.	If relevant, describes the class of input instances for the improved algorithm that would result in the best case or worst case running times.	

VCE Algorithmics (HESS): Unit 4 School-assessed Task 2017						
Assessment criteria	**Levels of Performance**					
	Not shown	**1–2 (low)**	**3–4**	**5–6 (medium)**	**7–8**	**9–10 (high)**
5. Understanding of the principles of computation		Outlines the goals of Hilbert's program, with some minor errors. Outlines some characteristics of a Turing machine, with some errors. Outlines the Halting problem, with some minor errors. Uses the term "decision problem" appropriately. Outlines some details of concepts relevant to how the equivalence of computational formalisms is shown, with some errors. Outlines the thought experiment that underpins the Chinese Room argument. Outlines some features of an alternative method of computation.	Describes briefly aspects of the goals of Hilbert's program and its outcome and its historical context, possibly with minor errors. Describes the components of a Turing machine. Executes one step of a Turing machine. Describes briefly some characteristics of decidable or undecidable problems. Describes briefly some concepts relevant to how the equivalence of computational formalims is shown. Describes the thought experiment that underpins Chinese Room argument and briefly describes aspects of its core position, standard replies, or historical context. Describes using everyday language the core features of an alternative model of computation.	Describes, with appropriate use of metalanguage, the goals of Hilbert's program, its outcome and its historical context and connection to the origin of computer science. Describes, with appropriate use of metalanguage, some aspects of the operation of a Turing machine. Executes a fixed number of steps of the Turing machine provided in either graphical or tabular format. Describes, with appropriate use of metalanguage, the concept of undecidability and a specific example of an undecidable problem, possibly with minor errors. Describes accurately and clearly concepts relevant to how the equivalence of computational formalims is shown.	Describes in detail the goals of Hilbert's program, its outcome, its historical context and connection to the origin of computer science. Describes in detail the operation of a Turing machine. Describes in detail the concept of undecidability. Describes coherently and in detail the characteristics a specific example of an undecidable problem. Explains in detail conceptually how the equivalence of computational formalisms is shown. Describes in detail and with appropiate use of metalanguage the Chinese Room argument, its historical context, motivation, and standard responses.	Explains comprehensively and precisely the goals of Hilbert's program, its outcome, its historical context and connection to the origin of computer science. Explains comprehensively and precisely the operation of a Turing machine. Explains comprehensively and precisely the concept of undecidability. Describes comprehensively and precisely the characteristics of a specific undecidable problem. Explains comprehensively and precisely conceptually how the equivalence of computational formalisms is shown. Describes comprehensively and precisely several standard responses to the Chinese Room argument. The standard responses are clearly distinguished, with no confusion or conflation.

VCE Algorithmics (HESS): Unit 4 School-assessed Task 2017						
Assessment criteria	**Levels of Performance**					
	Not shown	**1–2 (low)**	**3–4**	**5–6 (medium)**	**7–8**	**9–10 (high)**
				Describes the core position of Chinese Room argument, one standard response to the argument, and its historical context. Describes, with appropriate use of metalanguage, an alternative model of computation, including some aspects of its operation.	Describes in detail an alternative model of computation, including some technical aspects of the method and its operation.	Explains comprehensively and precisely an alternative model of computation, including significant technical details of the method and its operation.

VCE Algorithmics (HESS): Unit 3 School-assessed Task 2017						
Assessment criteria	**Levels of Performance**					
	Not shown	**1 – 2 (low)**	**3 – 4**	**5 – 6 (medium)**	**7 – 8**	**9 – 10 (high)**
6. Skills in the discussion and evaluation of computation concepts		Outlines the central premise of the Church-Turing thesis. Outlines some aspect of the consequences of undecidability. Outlines the relationship between Turing machines and computational complexity theory. Outlines the central premise of Cobham's thesis. Discusses in brief the strength of one of the standard responses to the Chinese Room argument. Outlines some aspects of the connections between the Chinese Room argument and artificial intelligence. Outlines the merits of an alternative method of computation.	Discusses in brief the Church-Turing thesis, its historical context, and limited aspects of its merits or limitations. Discusses in brief the implications of undecidability for Hilbert's Program. Demonstrates the undeciability of the Halting problem through the use of a poorly explained and limited argument. Discusses in brief the relationship between Turing machines and computational complexity theory. Discusses in brief Cobham's thesis and limited aspects of its merits or limitations. Evaluates the strength of at least two of the standard responses to the Chinese Room argument.	Discusses clearly the Church-Turing thesis, its historical context, and some aspects of its merits or limitations. Discusses clearly several implications of undecidability. Demonstrates the undecidability of the Halting problem through the use of a relevant and clear argument. Discusses clearly the relationship between Turing machines and computational complexity theory. Describes clearly Cobham's thesis and some of its merits and limitations. Compares the strength of two of the standard responses to the Chinese Room argument.	Evaluates the merits or limitations of the Church-Turing thesis and discusses its implications. Demonstrates the undecidability of the Halting probem through the use of an effective and detailed argument. Discusses in detail and coherently the relationship between Turing machines and computational complexity theory. Evaluates the merits or limitations of the Cobham's thesis and discusses its meaning for computational complexity theory. Formulates a position either for or against the Chinese Room argument through an evaluation of standard responses and connects this position to its implications for artificial intelligence.	Comprehensively and precisely evaluates the merits and limitations of Church-Turing thesis and discusses its implications. Demonstrates precisely the undecidability of the Halting problem through the use of a comprehensive and well structured argument. Discusses comprehensively and precisely the relationship between Turing machines and computational complexity theory. Comprehensively and precisely evaluates the merits and limitations of Cobham's thesis and discusses its meaning for computational complexity theory. Formulates a substantiated position either for or against the Chinese Room argument through a sound evaluation of several standard responses and connects this position to its implications for artificial intelligence.

VCE Algorithmics (HESS): Unit 4 School-assessed Task 2017						
Assessment criteria	**Levels of Performance**					
	Not shown	**1–2 (low)**	**3–4**	**5–6 (medium)**	**7–8**	**9–10 (high)**
			Discusses in brief the connections between the Chinese Room argument and artificial intelligence. Discusses in brief how an alternative method of computation might be used to overcome current limits of computation.	Discusses clearly the connections between the Chinese Room argument and artificial intelligence. Discusses clearly how an alternative method of computation might be used to overcome current limits of computation and some aspects of its merits and limitations.	Explains in detail how an alternative model of computation might be used to overcome current limits of computation and evaluates briefly its merits and limitations.	Explains comprehensively and precisely how an alternative model of computation might be used to overcome current limits of computation and evaluates its merits and limitations.

Authentication of VCE Algorithmics (HESS) School-assessed Task

Teachers are reminded of the need to comply with the authentication requirements specified in the Scored assessment: School-based Assessment section of the *VCE and VCAL Administrative Handbook 2017*.

Teachers must be aware of the following requirements for the authentication of VCE Algorithmics (HESS) School-assessed Task:

1. The body of work created for the School-assessed Task is based on work developed and completed in Unit 4 Outcome 1, Outcome 2 and Outcome 3.
2. The Authentication Record Form should be completed by the teacher and the students should be provided with feedback on their progress at each observation.
3. Undue assistance should not occur at any time during the development of the body of work. Teachers are reminded that it is not appropriate to provide detailed advice on, corrections to, or actual reworking of students' work.
4. The development and documentation of the student's thinking and working practices must be sighted and monitored throughout the unit to authenticate the work as the student's own. Students must acknowledge the source of materials and information used to support the development of their work.
5. Students should be encouraged to complete their work at school.
6. During development of the data model and solution teachers must plan and use observations of student work in order to monitor and record each student's progress as part of the authentication process. Teachers must ensure that all source and reference material, all use of non-school (home, outsourced) resources and any external assistance (for example, tutors) are acknowledged on the authentication form. If a student acknowledges using external resources or receiving external assistance, the teacher should record complete details as an attachment to the Authentication Form.
7. The authentication procedures must be followed for all student work in relation to the School-assessed Task. School-based Assessment audits include the inspection of authentication records.

Authentication Record Form VCE Algorithmics (HESS) 2017

Unit 4 School-assessed Task

This form must be completed by the class teacher. It provides a record of the monitoring of the student's work in progress for authentication purposes. This form is to be retained by the school and filed. It may be collected by the VCAA as part of its School-based Assessment audit.

Student name: ..... Student No

School: .....

Teacher: .....

Component of School-assessed Task	Date observed/ submitted	Authentication comments	Teacher's initials	Student's initials
Observation: Submission of written explanation Student has submitted the written explanation of formal analysis techniques and the practical limits of computability.				
Observation: Analysis of algorithm Student has analysed a naïve algorithm.				
Observation: Progressive development of data model Student has continued to develop a data model and documentation.				
Observation: Submission of written explanation Student has submitted the written explanation of algorithm design patterns and techniques for addressing the limits of computation.				
Observation: Development of algorithm Student has submitted an improved algorithm and analysis.				
Observation: Report Student has commenced the process of developing their report.				
Observation: Report Student has submitted an explanation of the universality of computation and algorithms.				

I declare that all resource materials and assistance used have been acknowledged and that all unacknowledged work is my own.

Student signature …… Date ……

2017

Victorian Certificate of Education
Algorithmics (HESS) Assessment Sheet
School–assessed Task: Unit 4

STUDENT NAME

Teachers need to make judgments on the student's performance for each assessment criterion. Teachers will be required to choose one number from 0–10 to indicate how the student performed on each criterion with comments, as appropriate.

STUDENT NUMBER

ASSESSING SCHOOL NUMBER

Assessment criteria

	Not Shown (0)	Low (1–2)	(3–4)	Med (5–6)	(7–8)	High (9–10)
The extent to which the student demonstrates:						
1 understanding of formal algorithm analysis	☐	☐	☐	☐	☐	☐
2 skills in establishing the efficiency of a naïve algorithm	☐	☐	☐	☐	☐	☐
3 understanding of advanced algorithm design	☐	☐	☐	☐	☐	☐
4 skills in developing an improved algorithm in response to a naïve algorithm	☐	☐	☐	☐	☐	☐
5 understanding of the principles of computation	☐	☐	☐	☐	☐	☐
6 skills in the discussion and evaluation of computation concepts	☐	☐	☐	☐	☐	☐

Levels of Performance: Teacher's Comments

You may wish to comment on aspects of the student's work that led to your assessment of High, Medium, Low,or Not Shown for specific criteria.

If a student does not submit the School–assessed Task at all, N/A should be entered here. ☐