VCE Software Development 2024
Unit 3 School-based Assessment

Video 4
Assessing the
Unit 3 Outcome 1 SAC

VICTORIAN CURRICULUM AND ASSESSMENT AUTHORITY
VICTORIA State Government

Acknowledgement of Country

The VCAA respectfully acknowledges the Traditional Owners of Country throughout Victoria and pays respect to the ongoing living cultures of First Peoples.

VCE Software Development 2024
Unit 3 School-based Assessment

Video 4
Assessing the
Unit 3 Outcome 1 SAC

Phil Feain
Digital Technologies Curriculum Manager
VCAA

VICTORIAN CURRICULUM AND ASSESSMENT AUTHORITY

VICTORIA State Government

Purpose of this presentation

- to build the capacity of teachers to develop compliant, rigorous and engaging VCE assessment tasks in line with the VCE assessment principles
- provide an overview of how to assess the Unit 3 Outcome 1 School-assessed Coursework (SAC) task.

Unit 3 Outcome 1

Unit 3 Outcome 1 – The outcome

On completion of this unit the student should be able to interpret teacher-provided solution requirements and designs, and apply a range of functions and techniques using a programming language to develop and test working software modules.

Key knowledge

Data and information

- characteristics of data types
- types of data structures, including associative arrays (or dictionaries or hash tables), one-dimensional arrays (single data type, integer index) and records (varying data types, field index)

Approaches to problem-solving

- methods for documenting a problem, need or opportunity
- methods for determining solution requirements, constraints and scope
- methods of representing designs, including data dictionaries, mock-ups, object descriptions and pseudocode
- formatting and structural characteristics of files, including delimited (CSV), plain text (TXT) and XML file formats
- a programming language as a method for developing working modules that meet specified needs
- naming conventions for solution elements
- processing features of a programming language, including classes, control structures, functions, instructions and methods
- algorithms for sorting, including selection sort and quick sort
- algorithms for binary and linear searching
- validation techniques, including existence checking, range checking and type checking
- techniques for checking that modules meet design specifications, including trace tables and construction of test data
- purposes and characteristics of internal documentation, including meaningful comments and syntax.

Key skills

- interpret solution requirements and designs to develop working modules
- use a range of data types and data structures
- use and justify appropriate processing features of a programming language to develop working modules
- develop and apply suitable validation, testing and debugging techniques using appropriate test data
- document the functioning of modules and the use of processing features through internal documentation.

Unit 3 Outcome 1 – The assessment task

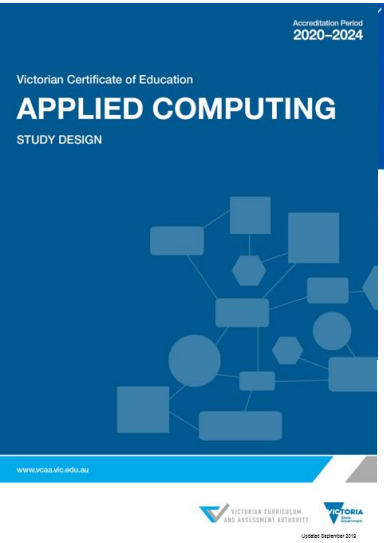
Contribution to final assessment

School-assessed Coursework for Unit 3 will contribute 10 per cent to the study score.

Outcomes	Marks allocated	Assessment tasks
Unit 3 **Outcome 1** Interpret teacher-provided solution requirements and designs, and apply a range of functions and techniques using a programming language to develop and test working software modules.	100	In response to teacher-provided solution requirements and designs, create working modules.
Total marks	100	

Assessing the Unit 3 Outcome 1 SAC task using VCAA resources

Unit 3 Outcome 1 Resources



VCE Applied Computing: Performance Descriptors

SOFTWARE DEVELOPMENT UNIT 3 OUTCOME 1
SCHOOL-ASSESSED COURSEWORK

Performance Descriptors

	DESCRIPTOR: typical performance in each range				
	Very low	**Low**	**Medium**	**High**	**Very high**
Unit 3 Outcome 1 ***Interpret teacher-provided solution requirements and designs, and apply a range of functions and techniques using a programming language to develop and test working software modules.***	Limited interpretation of solution requirements and designs to develop working modules.	Some interpretation of solution requirements and designs to develop working modules.	Sound interpretation of solution requirements and designs to develop working modules.	Most solution requirements and designs are interpreted accurately to developing working modules.	All solution requirements and designs are interpreted accurately to developing working modules.
	Limited selection and use of data types and data structures.	Some selection and use of appropriate data types and data structures.	Sound selection and use of data types and data structures to develop working modules.	Detailed selection of relevant data types and data structures to develop working modules.	Comprehensive selection of relevant data types and data structures to develop working modules.
	Limited selection and use of processing features of the programming language to develop some working modules.	Some selection and use of appropriate processing features of the programming language to develop some working modules.	Sound selection and use of appropriate processing features of the programming language to develop some working modules.	Most processing features of the programming language have been selected and used to develop all working modules.	Comprehensive selection and use of relevant processing features of the programming language to develop all working modules.
	Limited explanation of how the selected processing features are used to develop working modules.	Some justification and explanation of how the selected processing features are used to develop working modules.	Sound justification and explanation of how the selection of appropriate processing features are used to develop working modules.	Detailed justification and explanation of how the selection of appropriate processing features of the programming language are used to develop working modules.	Comprehensive justification and explanation of how the selection of appropriate processing features of the programming language are used to develop working modules.

Unit 3 Software Development – 2024
Outcome 1 Software development: programming – Developing a marking scheme – Sample

Outcome 1
On completion of this unit the student should be able to interpret teacher-provided solution requirements and designs, and apply a range of functions and techniques using a programming language to develop and test working software modules.

Developing a marking scheme – Marks allocated – 100
Refer to the key skills on the VCAA performance descriptors when developing a marking scheme for the assessment task. Determine the weighting of the marks out of 100 for each key skill or performance descriptor. When determining weightings consider the time that students will take to complete each task as well as the level of difficulty of each task. Marks should be allocated to ensure students can demonstrate a range of levels of performance in the task.

Key knowledge	Key skills	VCAA Performance descriptors (Very high)	
• methods for documenting a problem, need or opportunity • methods for determining solution requirements, constraints and scope • methods of representing designs, including data dictionaries, mock-ups, object descriptions and pseudocode	• interpret solution requirements and designs to develop working modules	• All solution requirements and designs are interpreted accurately to developing working modules.	Students are to interpret the solution requirements and designs for between three and six working modules. Possible number of marks – 10 marks
• characteristics of data types • types of data structures, including associative arrays (or dictionaries or hash tables), one-dimensional arrays (single data type, integer index) and records (varying data types, field index) • formatting and structural characteristics of files, including delimited (CSV), plain text (TXT) and XML file formats	• use a range of data types and data structures	• Comprehensive selection of relevant data types and data structures to develop working modules.	Students are to use a range of relevant data types and data structures within their software modules. Possible number of marks – 10 marks
• a programming language as a method for developing working modules that meet specified needs • naming conventions for solution elements • processing features of a programming language, including classes, control structures, functions, instructions and methods • algorithms for sorting, including selection sort and quick sort • algorithms for binary and linear searching	• use and justify appropriate processing features of a programming language to develop working modules	• Comprehensive selection and use of relevant processing features of the programming language to develop all working modules. • Comprehensive justification and explanation of how the selection of appropriate processing features of the programming language are used to develop working modules.	Students are to use appropriate processing features, naming conventions and sorting and searching algorithms to develop their software modules. A higher weighting of marks should be included to meet this key skill or performance descriptor. Possible number of marks – 40 marks Students are to justify and explain their selection of processing features and sorting and searching algorithms used to develop their working modules. Possible number of marks – 10 marks
• validation techniques, including existence checking, range checking and type checking • techniques for checking that modules meet design specifications, including trace tables and construction of test data	• develop and apply suitable validation, testing and debugging techniques using appropriate test data	• Comprehensive use of relevant data validation techniques are applied efficiently and effectively to check the reasonableness of all input data. • Comprehensive use of test data is expressed in a testing table, with both expected and actual output stated, and showing detailed evidence of debugging.	Students are to use and apply relevant data validation techniques to check all input data. Possible number of marks – 10 marks Students test their working modules using appropriate testing techniques. Possible number of marks – 10 marks
• purposes and characteristics of internal documentation, including meaningful comments and syntax	• document the functioning of modules and the use of processing features through internal documentation	• All software modules include comprehensive internal documentation regarding the functioning of modules and use of selected processing features.	Students are to include internal documentation within their working modules. Possible number of marks – 10 marks

Developing a marking scheme – Sample

Unit 3 Software Development – 2024 Outcome 1 Software development: programming – Developing a marking scheme – Sample			
Outcome 1 On completion of this unit the student should be able to interpret teacher-provided solution requirements and designs, and apply a range of functions and techniques using a programming language to develop and test working software modules.			**Developing a marking scheme – Marks allocated – 100** Refer to the key skills or the VCAA performance descriptors when developing a marking scheme for the assessment task. Determine the weighting of the marks out of 100 for each key skill or performance descriptor. When determining weightings consider the time that students will take to complete each task as well as the level of difficulty of each task. Marks should be allocated to ensure students can demonstrate a range of levels of performance in the task.
Key knowledge	**Key skills**	**VCAA Performance descriptors (Very high)**	
• methods for documenting a problem, need or opportunity • methods for determining solution requirements, constraints and scope • methods of representing designs, including data dictionaries, mock-ups, object descriptions and pseudocode	• interpret solution requirements and designs to develop working modules	• All solution requirements and designs are interpreted accurately to developing working modules.	Students are to interpret the solution requirements and designs for between three and six working modules. Possible number of marks – 10 marks
• characteristics of data types • types of data structures, including associative arrays (or dictionaries or hash tables), one-dimensional arrays (single data type, integer index) and records (varying data types, field index) • formatting and structural characteristics of files, including delimited (CSV), plain text (TXT) and XML file formats	• use a range of data types and data structures	• Comprehensive selection of relevant data types and data structures to develop working modules.	Students are to use a range of relevant data types and data structures within their software modules. Possible number of marks – 10 marks
• a programming language as a method for developing working modules that meet specified needs • naming conventions for solution elements • processing features of a programming language, including classes, control structures, functions, instructions and methods • algorithms for sorting, including selection sort and quick sort • algorithms for binary and linear searching	• use and justify appropriate processing features of a programming language to develop working modules	• Comprehensive selection and use of relevant processing features of the programming language to develop all working modules. • Comprehensive justification and explanation of how the selection of appropriate processing features of the programming language are used to develop working modules.	Students are to use appropriate processing features, naming conventions and sorting and searching algorithms to develop their software modules. A higher weighting of marks should be included to meet this key skill or performance descriptor. Possible number of marks – 40 marks Students are to justify and explain their selection of processing features and sorting and searching algorithms used to develop their working modules. Possible number of marks – 10 marks
• validation techniques, including existence checking, range checking and type checking • techniques for checking that modules meet design specifications, including trace tables and construction of test data	• develop and apply suitable validation, testing and debugging techniques using appropriate test data	• Comprehensive use of relevant data validation techniques are applied efficiently and effectively to check the reasonableness of all input data. • Comprehensive use of test data is expressed in a testing table, with both expected and actual output stated, and showing detailed evidence of debugging.	Students are to use and apply relevant data validation techniques to check all input data. Possible number of marks – 10 marks Students test their working modules using appropriate testing techniques. Possible number of marks – 10 marks
• purposes and characteristics of internal documentation, including meaningful comments and syntax	• document the functioning of modules and the use of processing features through internal documentation	• All software modules include comprehensive internal documentation regarding the functioning of modules and use of selected processing features.	Students are to include internal documentation within their working modules. Possible number of marks – 10 marks

Some do’s

- Take the time to develop the assessment task and develop a suitable marking scheme.
- Refer to the key skills and the performance descriptors.
- Consider the number of marks to be awarded.
- Consider the weighting of the marks for each component. This enables more marks for more complex and time consuming components of the assessment task and enables you to differentiate more between your stronger students and your weaker students.
- Ensure you have a range of levels of performance from very low to very high. Having marks in multiples of 5 helps you to separate the marks out for students.
- Ensure your marks add up to 100 marks.

Some don’ts

- Don’t just stick a copy of the VCAA Performance descriptors at the back of the assessment task. It does not break down how you are marking each component and how they contribute to 100 marks.
- Don’t have the number of marks out of 10 or 20 or 30 and then say you’ll multiply by however much to get a score out of 100. This does not allow your student scores to be separated out and will bunch your scores.
- Don’t just use a commercial marking scheme without checking it against your assessment task. Check to see that it meets the key skills and the performance descriptors and that the marks total to 100 marks.
- Don’t forget to go through the marking scheme with the students before they complete the assessment task. They should know what they are being assessed on and how they are being marked.

Contact

- **Phil Feain – Digital Technologies Curriculum Manager (VCAA)**
- **Ph: (03) 9059 5146**
- **Philip.Feain@education.vic.gov.au**

Authorised and published by the
Victorian Curriculum and Assessment Authority