VCE Software Development 2024
Unit 3 School-based Assessment

Video 3
Planning 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 3
Planning 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 plan 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

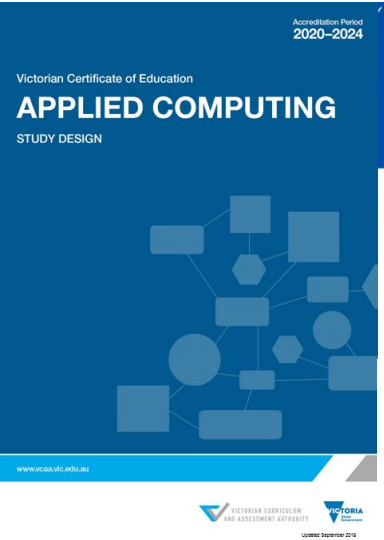
Contribution to final assessment

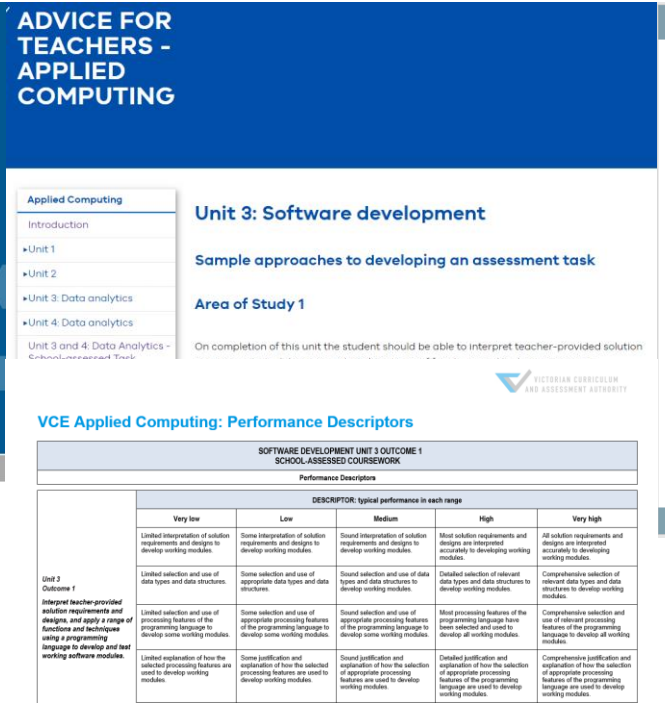
School-assessed Coursework for Unit 3 will contribute 10 per cent to the study score.

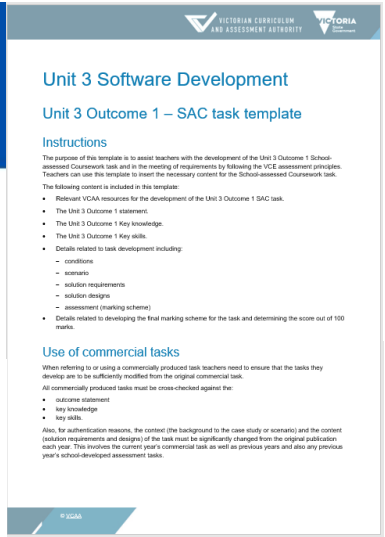
Outcomes	Marks allocated	Assessment tasks
Unit 3 **Outcome 1** Interpret teacher-provided solution requirements and designs, and apply a range of functions and techniques using a programming language to develop and test working software modules.	100	In response to teacher-provided solution requirements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

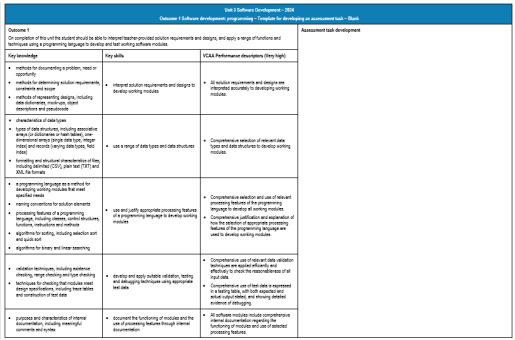
Planning the Unit 3 Outcome 1 SAC task using VCAA resources

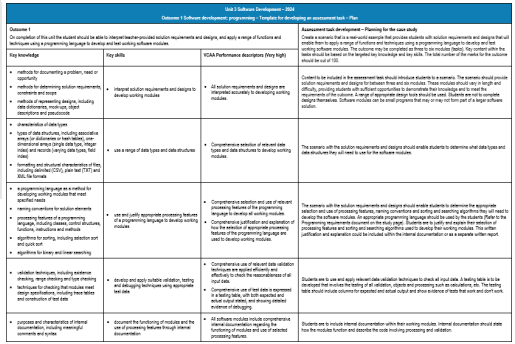
Unit 3 Outcome 1 Resources











VICTORIAN CURRICULUM AND ASSESSMENT AUTHORITY

VICTORIA State Government

Task development template – Blank

Unit 3 Software Development – 2024

Outcome 1 Software development: programming – Template for developing an assessment task – Blank

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.

Assessment task development

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.
• 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.
• 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.
• 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.
• 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.

Task development template – Plan

Unit 3 Software Development – 2024 Outcome 1 Software development: programming – Template for developing an assessment task – Plan			
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.			**Assessment task development – Planning for the case study** Create a scenario that is a real-world example that provides students with solution requirements and designs that will enable them to apply a range of functions and techniques using a programming language to develop and test working software modules. The outcome may be completed as three to six modules (tasks). Key content within the tasks should be based on the targeted key knowledge and key skills. The total number of the marks for the outcome should be out of 100.
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.	Content to be included in the assessment task should introduce students to a scenario. The scenario should provide solution requirements and designs for between three and six modules. These modules should vary in length and difficulty, providing students with sufficient opportunities to demonstrate their knowledge and to meet the requirements of the outcome. A range of appropriate design tools should be used. Students are not to complete designs themselves. Software modules can be small programs that may or may not form part of a larger software solution.
• 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.	The scenario with the solution requirements and designs should enable students to determine what data types and data structures they will need to use for the software modules.
• 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.	The scenario with the solution requirements and designs should enable students to determine the appropriate selection and use of processing features, naming conventions and sorting and searching algorithms they will need to develop the software modules. An appropriate programming language should be used by the students (Refer to the Programming requirements document on the study page). Students are to justify and explain their selection of processing features and sorting and searching algorithms used to develop their working modules. This written justification and explanation could be included within the internal documentation or as a separate written report.
• 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. A testing table is to be developed that involves the testing of all validation, objects and processing such as calculations, etc. The testing table should include columns for expected and actual output and show evidence of tests that work and don't work.
• 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. Internal documentation should state how the modules function and describe the code involving processing and validation.

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