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.

© VCAA