|  |
| --- |
| **SOFTWARE DEVELOPMENT UNIT 3 OUTCOME 1****SCHOOL-ASSESSED COURSEWORK** |
| **Performance descriptors** |
|  |  |
| **Unit 3****Outcome 1**On completion of this unit the student should be able to interpret teacher-provided solution requirements and designs and use appropriate features of an object-oriented programming language to develop working software modules. | **DESCRIPTOR: typical performance in each range** |
| **Key skill** | **Very Low** | **Low** | **Medium** | **High** | **Very high** |
| interpret solution requirements and designs | Identifies functional requirements. | Interprets functional and non-functional requirements. Some requirements are interpreted accurately to develop software modules. | Interprets functional and non-functional requirements and designs, including: * mock-ups
* data dictionaries.

Some requirements and designs are interpreted accurately to develop software modules. | Interprets functional and non-functional requirements and designs, including: * IPO charts
* pseudocode
* object descriptions.

Most requirements and designs are accurately interpreted to develop working software modules. | Interprets functional and non-functional requirements and designs. All requirements and designs are accurately interpreted to fully develop four working software modules. |
| use a range of data types, data structures and data sources | Identifies data types, including text, numeric and Boolean. | Uses data types for local variables.Outlines, in the internal documentation, why the selected data types were used. | Uses appropriate data types for global variables and data structures, including arrays and records.Explains, in the internal documentation, why the selected data types and data structures were used. | Uses appropriate data types, data structures and data sources, including plain text, delimited and XML files.Explains, in the internal documentation, why the selected data sources were used. | Uses a range of data types, data structures and data sources.Explains and justifies, in the internal documentation, why the selected data types, data structures and data sources were used. |
| use and justify appropriate features of an OOP language to develop working software modules | Develops software modules using:* instructions
* arithmetic operators.
 | Develops software modules using:* local variables and constants
* logical and conditional operators
* control structures of sequence and selection.
 | Develops software modules using: * global variables
* the control structure of iteration/repetition.

Identifies, in the internal documentation, the features used within the OOP language.Develops a GUI for at least one software module. | Develops working software modules using: * functions and methods
* sorting and searching algorithms.

Explains, in the internal documentation, the purpose of the features used within the OOP language.Develops a GUI for at least two software modules. | Develops working software modules using: * classes and objects.

Justifies, in the internal documentation, the use of all features within the OOP language. |
| develop and apply suitable naming conventions and validation techniques within modules | Identifies naming conventions for use within software modules.Identifies input data for validation. | Applies naming conventions to variables within software modules.Validates input data using one of the following checks:* existence
* type
* range.
 | Applies naming conventions to variables and interface controls within software modules.Validates input data using two of the following checks:* existence
* type
* range.

Some inconsistencies are present. | Applies naming conventions to variables, interface controls and code structures within working software modules.Validates data using all the following checks:* existence
* type
* range.

Minor inconsistencies are present. | Applies suitable naming conventions to all solution elements within working software modules.Validates all relevant input data and checks the reasonableness of all input data.No inconsistencies are present. |
| document the functioning of modules using internal documentation | Identifies the functioning of software modules. | Outlines the functioning of software modules using internal documentation. | Uses internal documentation to describe the functionality of software modules and the use of data.Displays evidence of code maintenance.Some issues with clarity exist. | Uses internal documentation to explain the functionality of software modules and the use of data and code structures.Uses placeholder comments within software modules for future development.Minor issues with clarity exist. | Uses internal documentation to explain and justify the use of all data, code structures and functionality of all software modules.Clear and concise. |
| develop and apply suitable debugging and testing techniques using appropriate test data | Lists test data for software modules.Syntax errors are present. | Outlines some test data for software modules in a testing table.The testing table is incomplete.Uses debugging statements to check functionality of software modules.Logic errors are present. | Identifies test data for software modules in a testing table with expected output stated.Identifies input data to test validation checks.Uses breakpoints to support debugging and testing.Minor logic errors are present. | Constructs test data for working software modules using testing tables.Documents test cases comparing both expected and actual output.Documents suggested actions for rectifying errors in the testing tables.Run-time errors are present. | Reports and displays evidence of test data for working software modules.Documents a complete list of actions taken for failed tests in a testing table that compares both expected and actual output.All modules function correctly. |

KEY to marking scale based on the Outcome contributing 100 marks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Very Low 1–20 | Low 21–40 | Medium 41–60 | High 61–80 | Very High 81–100 |