



National Certificate of Educational Achievement
TAUMATA MĀTAURANGA Ā-MOTU KUA TAEA

Exemplar for Internal Achievement Standard

Digital Technologies Level 2

This exemplar supports assessment against:

Achievement Standard 91896

Use advanced programming techniques to develop a computer program

An annotated exemplar is an extract of student evidence, with a commentary, to explain key aspects of the standard. It assists teachers to make assessment judgements at the grade boundaries.

New Zealand Qualifications Authority

To support internal assessment

	Grade: Excellence
1.	For Excellence, the student needs to use advanced programming techniques to develop a refined computer program. This involves: - ensuring that the program is a well-structured, logical response to the specified task - making the program flexible and robust - comprehensively testing and debugging the program. Full samples of student evidence are available in our online Learning Management System, Pūtake. The student has ensured that their program is a well-structured, logical response to the task. For example, they have used abstractions (where appropriate), functions to avoid repeated code, constants when a value never changes (as required), and variables of appropriate scope. The student has made the program flexible and robust. For example: - The code works for expected, unexpected, and boundary values. - It is easy to extend the code's functionality (e.g. a function has been used to check the menu choices, so it would be easy to update the menu to add another level or path). - Derived values have been used to iterate through a collection, instead of using hard-coded values. - A GUI has been used to limit invalid input, and/or other appropriate techniques (such as try/except) have been used to check for validity. - Derived values are returned properties or have been calculated from other values. The student has comprehensively tested and debugged the program. For example, they have supplied test plans and/or annotated screenshots showing that all program components and the final program have been tested to ensure they work correctly for expected, boundary, unexpected, and invalid cases. There is also evidence that they have used others to test their program throughout its development. The final program has been refined based upon testing.

	Grade: Merit
2.	For Merit, the student needs to use advanced programming techniques to develop an informed computer program. This involves: • documenting the program with appropriate names and comments that describe code function and behaviour • following common conventions for the chosen programming language • testing and debugging the program effectively to ensure that it works on a sample of both expected cases and boundary cases. Full samples of student evidence are available in our online Learning Management System, Pūtake. The student has documented the program with variable/module names and comments that describe code function and behaviour. Descriptive variable/module names have been used, e.g. the menu module has been called 'display_menu' and the list holding the values of the order, 'current order list.' The code has comments at key points describing code function and behaviour, e.g. '#module' for the main ordering sequence, which can be called to start or to cancel the order '(restart). The student has followed common conventions for their chosen programming language. For example, they have • used all lowercase variable names for code written in Python, and function definitions are placed before or after the main function. • followed layout conventions (e.g. whitespace between definitions), and indentation and/or bracketing conventions are also present. • used an automated tool to check that their code follows common conventions. The student has tested and debugged the program effectively, ensuring that it works on a sample of both expected and boundary cases, as required. They have provided evidence of testing components (and/or the final program) to confirm that it works correctly on a range of boundary cases, e.g. for lives at 0 and 1, or menu selection at the upper and lower bounds of the choices. The testing methodology is effective within the context of the problem.

	Grade: Achieved
3.	For Achieved, the student needs to use advanced programming techniques to develop a computer program. This involves: • writing code for a program that performs a specified task • using advanced techniques in a suitable programming language • setting out the program code clearly and documenting the program with comments • testing and debugging the program to ensure that it works on a sample of expected cases. Full samples of student evidence are available in our online Learning Management System, Pūtake. The student has written code for a program that performs a specified task. For example, this program meets all specified task requirements for a pizza ordering system. It allows users to enter typical data, and it outputs expected results. The student has used advanced techniques in a suitable programming language. For a pizza ordering program, this could include variables storing two different data types (e.g. string for name and boolean for delivery), an iteration control structure (e.g. a loop that repeats the entry prompt), selection (e.g. condition based on pick-up or delivery), input from the user and output of the order, and two or more advanced techniques from Explanatory Note 5, such as: • functions/procedures/methods for the ordering process and a menu display using parameters • event handling code to respond to events from the graphical user interface (GUI), and/or • a collection that is updated for the menu and the order. The student has set out the program code clearly and documented the program with some comments. The student has tested and debugged the program to ensure that it works on a sample of expected cases. For example, they documented their test cases using a screen capture and voiceover, explaining what tests were passed and what changes they needed to make. The program functions on expected input but may crash on boundary or invalid input.