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

Exemplar for Internal Achievement Standard

Digital Technologies Level 3

This exemplar supports assessment against:

Achievement Standard 91906

Use complex 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 complex programming techniques to develop a refined computer program. This involves: • ensuring that the program is a well-structured, logical response to the 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 the program is a well-structured, logical response to the task by using: • abstractions where appropriate • functions to avoid repeated code • constants as required when a value never changes • 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 functionality of the code (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 • they have used the GUI to limit invalid input, or used other appropriate techniques such as try/except to check for validity • derived values are returned properties, or are 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 the program components and final program have been tested to ensure that it works correctly for expected cases, boundary and unexpected or invalid cases • used others to test their program throughout the development process and refined the final program based upon testing.

	Grade: Merit
2.	For Merit, the student needs to use complex programming techniques to develop an informed computer program. This involves: • documenting the program with appropriate variable/module names and organised comments that describe code function and behaviour • following conventions for the chosen programming language • testing and debugging the program in an organised way to ensure that it works on a sample of both expected cases and relevant 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 organised comments that describe code function and behaviour. For example: • they use descriptive variable and module names, e.g. the menu module has been called ‘display_menu’, the list holding the ingredients of the planned menu ‘planned ingredient list’ • the code has comments at key points which describe code function and behaviour, e.g. ‘#function to display menu choices selected for the week – called from update choices or new week’. The student has followed common conventions for the chosen programming language. For example: • they used all lower-case variable names and CapWords for classes, for code written in Python • function definitions are placed before or after the main function, as per the programming language • layout conventions are followed, e.g. whitespace between definitions • indentation and/or bracketing conventions are followed as per the programming language • the student has used an automated tool to check that their code follows common conventions. The student has tested and debugged the program effectively to ensure that it works on a sample of both expected and relevant boundary cases. The program should provide the opportunity to test boundary cases. For example: • they have provided evidence of testing relevant components (or the complete program) to confirm that it works correctly on a range of boundary cases, e.g. the upper and lower bounds of the number of servings required • the testing methodology is effective within the context of the problem.

	Grade: Achieved
3.	For Achieved, the student needs to use complex programming techniques to develop a computer program. This involves: • writing code for a program that performs a specified task • using complex 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, the program meets all the specified task requirements for a menu planning system and allows users to enter typical data and outputs expected results. The student has used complex techniques in a suitable programming language. For example, for a menu planning program: • variables that store two different data types (e.g. string for ingredients and integer for quantity) • iteration control structure (e.g. a loop that repeats the entry prompt) • selection (e.g. condition based on meal type) • input from the user and output of the menus • and two or more complex techniques from Explanatory Note 5, such as: o programming or writing code for a graphical user interface (GUI) o reading from, or writing to, files or other persistent storage o object-oriented programming using class(es) and objects defined by the student o using types defined by the student o using third party or non-core API, library, or framework o using complex data structures (e.g., stacks, queues, trees). The student has set out the program code clearly and documented the program with comments. For example, most of the variable names are clear and sensible and the code includes some comments stating what the code does. The student has tested and debugged the program to ensure that it works on a sample of expected cases. They have also provided evidence of testing their program. The program works on expected input, but may crash on boundary or invalid input.