

**Internal Assessment Resource**

**Digital Technologies | Hangarau Matihiko Level 1**

This resource supports assessment against Achievement Standard 91883[[1]](#footnote-1)

**Standard title:** Develop a computer program

**Credits:** 4

**Resource title:** Quiz Quest

**Resource reference:** Digital Technologies | Hangarau Matihiko 1.7A Version 2

|  |
| --- |
| This resource:   * Clarifies the requirements of the achievement standard * Supports good assessment practice * Should be subjected to the school’s usual assessment quality assurance process * Should be modified to make the context relevant to students in their school/kura environment and ensure that submitted evidence is authentic |

|  |  |
| --- | --- |
| Date version published by Ministry of Education | November 2019 Version 2  To support internal assessment from 2020 |
| Authenticity of evidence | Teachers/Kaiako must manage authenticity for any assessment from a public source, because students may have access to the assessment schedule or student exemplar material.  Using this assessment resource without modification may mean that students’ work is not authentic. The teacher may need to change figures, measurements or data sources or set a different context or topic to be investigated or a different text to read or perform. |

**Internal Assessment Resource**

**Achievement standard:** 91883

**Standard title:**  Develop a computer program

**Credits:** 4

**Resource title:** Quiz Quest

**Resource reference:** Digital Technologies | Hangarau Matihiko1.7A Version 2

**Teacher/Kaiako guidelines**

The following guidelines are supplied to enable teachers/kaiako to carry out valid and consistent assessment using this internal assessment resource.

Teachers/kaiako need to be very familiar with the outcome being assessed by the achievement standard. The achievement criteria and the explanatory notes contain information, definitions, and requirements that are crucial when interpreting the standard and assessing students/ākonga against it.

**Context/Te Horopaki**

This activity requires students to create a quiz on a topic of their choice. For example - a quiz on the road code, a study aid for science or general knowledge or a language quiz to help people to learn Te Reo. The quiz should ask the user for their age or year level to determine that the quiz is suitable for them to use.

Students will construct a program that presents the quiz to the user, checks the answer and gives some kind of feedback. This open task allows for a number of different solutions of various complexity allowing for students to demonstrate their programming skills at an appropriate level. Examples include; a quiz using numbers, a simple quiz using lists of questions and answers, multiple choice quizzes or multi-player quizzes. The program must be clearly commented, tested and debugged so that it works as expected. A suitable programming language should be chosen that allows the student to use simple indexed data structures and/or functions/procedures to create a well-structured program.

Students are encouraged to adopt an iterative approach to develop their program. For example; break the overall task down into smaller achievable parts, code, test and evaluate each part before moving on to the next progressive improvement. They should be saving different working versions as they create their solution.

**Note:** A computer program will include variables of at least two different types, and sequence, selection, and iteration control structures. The program must also include one or more of: data stored in collections (e.g. lists, arrays dictionaries); user-defined methods, functions or procedures. Before beginning this task, teachers must ensure students are familiar with these concepts and the various programming conventions for their chosen programming language.

To test a program in an organised way students should be encouraged to think about how they will test their program for expected, boundary and unexpected cases. It is often useful to note down what they want to test and provide evidence of testing.

**Conditions/Ngā Tikanga**

Where a group approach is used, the teacher/kaiako needs to ensure that there is opportunity for each student to provide evidence for all aspects of the standard.

***Project based learning/collaboration***

Another way of collaborating is for a class to brainstorm approaches to the problem together (e.g. what collections of data to use and the implications of using that type of collection) as long as more than one approach is discussed. Students can then work on their own individual programs or create parts of a much bigger program together assuming they are all able to demonstrate the requirements of the standard within their part.

The 4 credits for the achievement standard indicates that approximately 40 hours needs to be allocated for teaching, learning (in and out of the classroom) and assessment in a program of study.

Teachers may want to combine this assessment with one for DT|HM achievement standard 91884 *Use basic iterative processes to develop a digital outcome* allowing for a greater time allocation for both assessments to be scheduled within your overall program.

You may want to give students guidance on the type of quiz that they intend to make in order to ensure it is appropriate to their programming skills and is of sufficient rigour to allow them to meet the standard.

Conditions of Assessment related to this achievement standard can be found at <http://ncea.tki.org.nz/Resources-for-Internally-Assessed-Achievement-Standards>

**Resource requirements/Ngā Rauemi**

Students will need access to a programming environment and relevant reference material for that programming language.

**Additional information/He Kōrero Atu**

**Coding conventions** are a set of guidelines for a specific programming language that recommend programming style, practices, and methods for each aspect of a program written in that language. These conventions usually cover file organisation, indentation, comments, declarations, statements, white space, naming conventions, programming practices, programming principles, programming rules of thumb, architectural best practices, etc.

**Internal Assessment Resource**

**Achievement standard:** 91883

**Standard title:**  Develop a computer program

**Credits:** 4

**Resource title:** Quiz Quest

**Resource reference:** Digital Technologies | Hangarau Matihiko1.7A Version 2

**Student/Ākonga instructions**

**Introduction/Kupu Arataki**

This assessment activity requires you to create a quiz program in a programming language and include questions on a topic of your choice.

You are going to be assessed on how well you structure and refine your program. A well-structured, refined program will have code that is easy to follow, update and maintain, and is free from bugs.

You may work with others while discussing potential problem solutions or generating your questions, but all of your program must be entirely constructed and written by you.

Teacher note: Insert due dates and timeframes

**Task/Hei Mahi**

Decide on the purpose for your quiz. For example, a quiz to help the user to study the road code or a study aid for an upcoming exam in another subject or a quiz to help your classmates learn Te Reo.

Decide on the style of quiz that you will make, keeping in mind the time you have available, your programming ability and the software and hardware resources available to you. Examples of quiz styles are listed below:

* A multiple choice quiz
* A word answer quiz
* A timed quiz
* A multi-player quiz
* A game format quiz
* Any other style of quiz.

Your quiz must allow for user input in some form.

**You need to think about:**

* *How your quiz begins.* For example, should you have many levels? Is your quiz only appropriate for a certain age range?
* *How to store your data*. What variables will you require and what type of data will your variables store (e.g. text, numeric, Boolean). Will you store data in collections (e.g. lists, arrays or dictionaries) to improve the structure, flexibility and robustness of your program?
* *How to structure your program.* What procedural structure will your program require? Will you create functions/method/procedures to improve the structure, flexibility and robustness of your program?
* *How you will give feedback to the user.* This could be in the form of a text prompt, a correct answer should the user answer incorrectly, a score that tracks user progress or any other method that you see fit to use. Also consider if you want to give immediate feedback, or only after quiz is complete.

**You must:**

* Create this quiz in an iterative manner. For example, break down the program into smaller achievable parts then code, test, debug and evaluate each part before moving on the next progressive improvement. It is advisable to save each version of the code as you make progress.

**Note:** To test a program in a comprehensive way, you should think about how you will test the program for various cases such as expected, boundary and unexpected input, what happens when lives run out or when a sprite gets to the border of the window. It is often useful to note down what you want to test and what you expect to happen, as well as what actually happened.

* Ensure that you comment your code appropriately as you develop it and use variable names and comments that describe code function and behaviour.
* Ensure that you have followed conventions for the programming language of your choice and that you have chosen a well-structured, logical response to the task.
* You should ensure that your code is robust and that it handles expected, boundary and invalid values.
* Wherever possible you should try to ensure that your code has a flexible structure to allow for continued development.

**Note:** Your quiz program must include variables of at least two different types, and sequence, selection, and iteration control structures. The program must also include one or more of: data stored in collections (e.g. at least 2 lists, arrays, dictionaries); OR at least 2 user-defined methods, functions or procedures.

**Working together:**

If you are creating your quiz as a group project you need to ensure that each member of the group is responsible for a particular part of the quiz and that each member can demonstrate their own skill at creating a well-structured program. One possible way to do this is to have many levels with different types of quizzes at each level, but a common theme across levels. Each team member could then create a level. In this case, you would need to discuss and collaborate on common variables, collections of data and/or functions and clearly show which team member created which part of the program.

**Assessment schedule/Mahere Aromatawai: Digital Technologies | Hangarau Matihiko** **91883 – Quiz Quest**

|  |  |  |
| --- | --- | --- |
| **Evidence/Judgements for Achievement/Paetae** | **Evidence/Judgements for Achievement with Merit/Kaiaka** | **Evidence/Judgements for Achievement with Excellence/Kairangi** |
| The student has developed a computer program which involves:   * writing a basic program that performs the specified task, using a suitable programming language   **For example (partial evidence)**  The student has written a simple, functional quiz program in the language of their choice. The program may not be structured very well.  The student has written the main code in a procedural manner, including use of sequence, iteration and conditions.  They have used two variables, one to keep track of the score and the other for the player name. They have used two lists to store the questions and answers OR they have used two functions/methods to interact with the user.  The student has used data stored in collections or user-defined methods, function or procedures.   * setting out the program code clearly and documenting the program with comments   **For example (partial evidence)**  Comments are present, but may not be particularly descriptive or frequent. The code has made use of indention, making it clear where each new section starts, e.g. # this code performs the quiz.   * testing and debugging the program to ensure that it works on a sample of expected cases.   **For example (partial evidence)**  The student has provided evidence of expected cases that were used to test and debug the program to show that the program works when the user inputs data that is expected.  Testing may be trial and error rather than clearly thought out.  The student has noted some expected test cases and has tested what happens when the program is run. Such testing may be observed and recorded by the teacher, presented in table form with minimal notes, recorded using a screen cast or described using examples.  *The examples above are indicative samples only* | The student has developed an informed computer program which involves:   * documenting the program with variable names and comments that describe code function and behaviour   **For example (partial evidence)**  The student has frequent clear comments throughout the code that help to describe relevant functions or sections of code.  The variable names clearly describe the data they hold.  e.g. # this function tests that the user input is a number. It will continue to ask for input until the input is a number.   * following conventions of the chosen programming language   **For example (partial evidence)**  The student has followed most common programming conventions for their chosen language.  Python files and functions contain a docstring explaining the purpose of the program/function. Constants are ALL\_CAPS with underscores separating the words if required.  Variable names use underscore rather than Camel case.  Functions appear before the main section and the main section of code is all at bottom, not between the functions, thus making the program easier to read.   * testing and debugging the program in an organised way to ensure that it works on a sample of both expected and relevant boundary cases.   **For example (partial evidence)**  The student tests frequently during development (observed and recorded) and the final program works when the user inputs data that is expected and checks or handles when the data is outside of specific thresholds. The student has not relied on trial and error approach and has decided upon expected and boundary cases that need to be tested.  Such testing may be observed and recorded by the teacher, presented in table form, recorded using a screen cast or described using examples.  In a situation where a user is asked to “choose the answer 1,2,3 or 4”, the program checks that the input is above 0 and under 5  The program might not correctly handle unexpected data and fail when a word is typed in where a number was expected.  *The examples above are indicative samples only* | The student has developed a refined computer program which involves:   * ensuring that the program is a well-structured, logical solution to the task   **For example (partial evidence)**  The student’s final program consists of multiple general purpose functions/methods so that the program flow is clear and there is no unnecessary or redundant code. Functions/methods are well named so that they are self-documenting.  OR – the student has used parallel data structures (lists/arrays) or a dictionary to store the quiz data.  Function to test input is an integer is named is\_num.  The code is clean, concise, and easily readable. The main program may be short and consists of multiple reusable user defined functions which do most of the logic and processing.  OR The quiz questions and answers are stored in parallel arrays to make checking for the correct answer more efficient.   * making the program flexible and robust   **For example (partial evidence)**  The student has created a program with a flexible structure through the use of functions/methods or data stored in collections.  The program uses actions, conditions and control structures effectively and without unnecessary repetition.  The program uses constants, variables and derived values in place of literals. Such variables appear at the top of the code where they can easily be seen and changed should those values change over time.  The program correctly handles expected, boundary and invalid values.  The student has named QUESTION\_VALUE as a constant value rather than using 5 (or whatever value a correct answer rates).  The student uses the length of the list, instead of a hard coded value to loop through questions and check for correct answers.  For example: They have used appropriate error handling techniques to ensure that program handles even unexpected cases or inputs.   * comprehensively testing and debugging the program   **For example (partial evidence)**  The student makes sure the program checks the validity of input data and deals with expected, boundary and unexpected or invalid data. All cases have been tested including testing each condition within their code.  Testing has been done in a systematic way. Test cases have been well-thought out and there is evidence or notes showing that the code works as expected for all cases.  *The examples above are indicative samples only* |

Final grades will be decided using professional judgement based on a holistic examination of the evidence provided against the criteria in the achievement standard.

1. This achievement standard is derived from both *The New Zealand Curriculum* and *Te* *Marautanga o Aotearoa.* [↑](#footnote-ref-1)