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

Internal Assessment Resource

Digital Technologies & Hangarau Matihiko Level 2

This resource supports assessment against Achievement Standard 91896[1]

Standard title:	Use advanced programming techniques to develop a computer program
Credits:	6
Resource title:	"Take Away" Ordering Program
Resource reference:	Digital Technologies & Hangarau Matihiko 2.7B

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	December 2018 Version 1 To support internal assessment from 2019
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/ākonga 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.

[1]*Achievement Standard 91896 is derived from both The New Zealand Curriculum and Te Marautanga o Aotearoa.*

Internal Assessment Resource

Achievement Standard:	91896
Standard title:	Use advanced programming techniques to develop a computer program
Credits:	6
Resource title:	“Take Away” Ordering Program
Resource reference:	Digital Technologies & Hangarau Matihiko 2.7B

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 the student to develop a refined computer program, using advanced programming techniques, that will process phone orders for a “take away” food business. The specific context of this resource is a pizza business.

Note: Teachers are encouraged to edit this default task to make it suitable for their community. This task can be modified for other take away food scenarios, such as sandwiches or smoothies from the school canteen, or take away coffee orders.

Conditions/Ngā Tikanga

It is recommended that students should have at least two identified checkpoints with their teacher as they work through this assessment activity to ensure they have an opportunity to ask questions and gather feedback.

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. The language chosen must support the required data types, control structures, advanced techniques and have good comment facilities.

They will also require software to provide evidence of their testing process. This could be word processing, slideshow, screencast or project management software.

Internal Assessment Resource

Achievement standard:	91896
Standard title:	Use advanced programming techniques to develop a computer program
Credits:	6
Resource title:	“Take Away” Ordering Program
Resource reference:	Digital Technologies & Hangarau Matihiko 2.7B

Student/Ākonga instructions

Introduction/Kupu Arataki

This assessment activity requires you to develop and test a computer program to process phone orders for a take away pizza business, using advanced programming techniques.

You will be assessed on how effectively you refine your program to ensure that the program:

- is a well-structured, logical response to the task
- has been comprehensively tested and debugged
- is flexible and robust.

You should consider these examples of ways of making a program flexible and robust:

- using actions, conditions, control structures and methods, functions, or procedures effectively
- checking input data for validity
- correctly handling expected, boundary and invalid cases
- using constants, variables and derived values in place of literals.

When developing your program, you must ensure your program:

- uses variables storing at least two types of data (e.g. numeric, text, Boolean)
- uses sequence, selection and iteration control structures
- takes input from a user, sensor(s), or other external source(s)
- produces output
- follows common conventions of the chosen programming language
- is documented with appropriate variable/module names and comments that describe the code function and behaviour.

Your program must use two or more advanced programming techniques.

Examples of advanced programming techniques include writing code that:

- modifies data stored in collections (e.g. lists, arrays, dictionaries)
- defines and manipulates multidimensional data in collections
- creates methods, functions, or procedures that use parameters and/or return values
- responds to events generated by a graphical user interface (GUI)
- requires non-basic string manipulation
- uses functionality of additional non-core libraries.

Teacher note:

Checkpoint 1: 16rd March
Checkpoint 2: 23rd March
Checkpoint 3: 30th March
Final Due Date: 8th April

Task/Hei Mahi

Scenario:

Dream Pizzas wants to computerise their phone orders. Specifically, they want to be able to enter customer details, pizza(s) ordered and pick-up or delivery requirements into a computer and have it display the delivery details, itemised order, and total cost. Phone orders generally consist of several kinds of pizza.

Your program must meet the following specifications:

- The program contains options for the phone operator to specify whether the pizza order is for pickup or delivery.
- If the order is for delivery:
  - the program should collect the customer's name, address and phone number
  - a $3 delivery charge should be added to the total cost.
- If the order is for pick up:
  - the program should ask the phone operator to enter the customer's name.
- The program should allow the phone operator to input how many pizzas the customer would like (maximum 5).
- A menu of at least 12 pizza names should be presented to the phone operator.
- Each pizza to be ordered should be selected from the choices available on the menu and the order information should be stored.
- The cost of the first seven (regular) pizzas on the menu is $8.50 and the rest are $5 more as they are gourmet pizzas.
- When the order is finished:
  - the names of ordered pizzas and their individual prices should be displayed
  - the total cost of the order, including any delivery charge, should be displayed
  - customer name should be displayed
  - if the pizza is for delivery the address and phone number should be displayed.
- The program should allow the operator to cancel the order.
- After the order information has been displayed the program should be ready to accept another order or exit.

You need to think about:

- How will you program and present the menu and receive user input?
  Will you use a GUI and write event-handling code, or will you use a text-based menu and typed user input?
- How will you 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 will you 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? What parameters and/or return values would be required?

- How will you validate input and give feedback to the user?
  What methods will you use to restrict and/or validate input. When will the program display output to the user?

Development:

- You should break the program up into components. Think about what information each component will need to do its job, and what information it will pass on to the rest of the program. Code, test and debug each component separately. As you complete each section, you should save your code with a new version number.

  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. It is often useful to note down what you want to test and what you expect to happen, as well as what actually happened. Testing can be demonstrated by making a brief screencast showing the program being comprehensively tested. If desired, you can take screenshots of your screencast and annotate them.
- Ensure that you comment your code appropriately, as you develop it and use variable/module names and comments that describe code function and behaviour.
- Ensure that you have followed conventions for the programming language of your choice and that your program is a well-structured, logical response to the task.
- You should ensure that your code is robust and that it handles expected, boundary and invalid cases.
- Wherever possible you should try to ensure that your code has a flexible structure to allow for continued development.

Assessment schedule/Mahere Aromatawai: Digital Technologies & Hangarau Matihiko 91896 – "Take Away" Ordering Program

Evidence/Judgements for Achievement/Paetae	Evidence/Judgements for Achievement with Merit/Kaiaka	Evidence/Judgements for Achievement with Excellence/Kairangi
Use advanced programming techniques to develop a computer program. The student has: ● written code for a program that performs a specified task **For example (partial evidence):** The program meets all of the specified task requirements for the pizza ordering system. The program includes: – variables that store two different data types (e.g. string for name and Boolean for delivery) – 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, such as: • functions/procedures/methods for the ordering process and menu display that take input parameters • event handling code to respond to events from the GUI • a collection for the menu and the order. ● set out the program code clearly and documented the program with comments	Use advanced programming techniques to develop an informed computer program. The student has: ● documented the program with variable/module names and comments that describe code function and behaviour **For example (partial evidence):** The student uses descriptive variable and module names, e.g. the menu module might have been called 'display_menu', the list holding the values of the order 'current order list'. The code has comments at key points which describe code function and behaviour, e.g. '#module' for the main ordering sequence which can be called to start or to cancel the order (restart). ● followed common conventions for the chosen programming language **For example (partial evidence):** The student uses all lower case variable names 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. ● tested and debugged the program effectively to	Use advanced programming techniques to develop a refined computer program. The student has: ● ensured that the program is a well-structured, logical response to the task ● made the program flexible and robust **For example (partial evidence):** The student has used abstractions where appropriate. Functions have been used to avoid repeated code. 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 or delete another pizza item). They have used derived values (e.g. length(pizzas) to iterate through a collection instead of hard coded values. The code works for expected, unexpected and boundary values. They have used the GUI to limit invalid input or used other appropriate techniques such as try/except to check for validity. Constants are used as required when a value never changes e.g. delivery charge, max order size. Derived values are returned properties or are calculated from other values e.g. students have determined the size for the pizza order collection (e.g. array) based on the user input for number of pizzas ordered. Students have used variables of appropriate scope (e.g. the variable storing the total value of

For example (partial evidence): Most of the variable names are sensible and the code includes some comments but these comments don't describe the code's function Eg # order checker ● tested and debugged the program to ensure that it works on a sample of expected cases **For example (partial evidence):** Student has provided evidence of testing their program. The testing has been completed for expected cases, but the program may have bugs when boundary or invalid data is entered. *The examples above are indicative samples only*	ensure that it works on a sample of both expected and relevant boundary cases. **For example (partial evidence):** The student has evidence of testing and corresponding debugging of their program to confirm that it works correctly on a range of values, including boundary values. They have documented the development of their program using a testing log or version updates. Evidence has been collected to show that the program reacts correctly to the number of pizzas ordered, and the testing debugging ensured the program works on boundary cases, e.g. the program has been tested on order sizes of 0, 1, 5 and 6. *The examples above are indicative samples only*	the pizza order should be declared in that module and not as a global). ● comprehensively testing and debugging the program. **For example (partial evidence):** The student has supplied test plans and/or annotated screenshots/a screenshot 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. They have used others to test their program throughout the development process and have refined their final program based upon testing. *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.