# Internal Assessment Resource

| | |
|---|---|
| **Achievement standards:** | 91906 and 91907 |
| **Standard title:** | 91906 Use complex programming techniques to develop a computer program (6 credits)<br>91907 Use complex processes to develop a digital technologies outcome (6 credits) |
| **Credits:** | 12 |
| **Resource title:** | Developing for Purpose |
| **Resource reference:** | Digital Technologies & Hangarau Matihiko 3.7A_3.8A |

## Student/Ākonga instructions

## Introduction/Kupu Arataki

This assessment activity requires you to plan, develop and create a complex computer program.

You will be assessed on

- how effectively you use project management tools and techniques to plan and manage the development of a digital outcome
- how effectively you decompose the problem into smaller components, and test and refine your media outcome so that it is a high-quality response to the task
- how well you have addressed relevant implications
- how well you synthesise information from the planning, testing and trialling of components to develop a high-quality response to the task (e.g. well-structured, logical, flexible, robust and comprehensively tested program)
- discuss how this information assisted in the development of a high-quality outcome.

Due date: Friday 18 August 2023

## Task/Hei Mahi

### Starting point ideas:

**An ordering system for a take-away food shop**. A GUI menu takes input from the operator (menu items are chosen from a series of drop-down lists, including quantity) and interacts with an array of food items. Each order has a space for entering the name associated with the order. The program stores orders in an array. The program allows for take-away and delivery orders and calculates the final cost of the order. The menu drives all aspects of the ordering system.

**A quiz program for testing knowledge of a particular topic**. A GUI based menu interacts with an array of questions. The questions are multiple choice. A record of correct/incorrect answers is kept, and statistics gathered about the responses. The menu allows for users to skip and retry questions as well as resetting the whole quiz. Multiple arrays are used to hold questions - attempted and not-attempted. The GUI includes drop-down lists and at least one of radio buttons or text-boxes.

Other projects that you may wish to consider as starting points are:

**A sports analysis program** that takes GUI input of events/times/scores etc during a match or game or even series of games and allows the user to analyse the sports results.

**A GUI resource-tracking program** that tracks what items have been given/borrowed to whom and when they are due for return. The program should be able to query the array of items for what is lent out, what is still in etc. This is a generic library program suitable for any type of resource.

**A GUI based menu-planning program for a school camp** that allows users to plan a menu from a selection of common foods and determines if the menu is healthy or not. The program will need to use the calorie/KJ value of foods and compare it against 'healthy' criteria (e.g. total energy per day, contains vegetables etc). It will also need to be able to modify a menu if it does not meet the healthy criteria.

**Find nearest emergency service.** This program would have coordinates stored for different emergency service buildings (fire, ambulance, police, fast food?) and when a user enters their coordinates (e.g. using a smartphone to be able to accept the input) and asks for a service, it would show the closest building for dispatch. Coordinates can be obtained from Google maps.

**Gene sequence search program** that does a linear search through a gene sequence for a particular set of characters. The program could count how many occurrences there are in the substring and where they occur. A key to gene sequence files can be found at http://www.genomatix.de/online_help/help/sequence_formats.html and example gene sequence files can be found at http://www.ncbi.nlm.nih.gov/genbank/samplerecord/ (scrolling to the bottom will reveal the gene sequence).

You may use the above ideas as starting points for your own project or you may come up with a unique idea of your own. **If you cannot come up with a project within two lessons, a project may be given to you.** Your idea, and the context it works in (i.e. how it is unique to you) must be discussed and agreed upon by your teacher before you start the work outlined in the planning section, below.

## Program requirements

Writing code that performs a specified task and uses complex programming techniques. You should be able to show evidence of at least **TWO** complex programming techniques. Examples of complex programming techniques include writing code that:

- creates a graphical user interface (GUI)
- reads from, or writes to, files or other persistent storage
- defines class(es) and creates objects
- defines and uses custom type(s)
- uses third party or non-core API, library or framework
- uses complex data structures (e.g. stacks, queues, trees).

Programming code should be set out clearly. Document the program with appropriate variable/module names and organised comments that describe code function and behaviour. Use appropriate variable/module names and follow conventions for your chosen programming language.

- Show comprehensive testing and debugging of the program. This should be carried out in an organised way to ensure that it works on expected cases and relevant boundary cases. (This could possibly be shown in your versioning, or through screencasts.)
- Ensure that the program is a well-structured, logical response to the task.
- Ensure that the program is flexible and robust.

## Planning, Development & Testing Requirements

Your program needs to be developed using complex processes and you will need to show evidence of this process.

- Use recognised project management tools to help you plan and the development of your program.
- Decompose your outcome into components. You should trial multiple components or techniques to determine which one will be best for the overall quality of your outcome. For example, you might trial the use of pop-up menus, text boxes or radio-buttons to get the same information from the user in a GUI. You might assess their quality in terms of functionality, usability, flexibility, interface aesthetic, etc. for your outcome.
- You need to provide evidence of effectively using the information from testing and trialling to improve the functionality of the outcome. You may use your project management tools to record and gather evidence of testing, trialling and feedback. Alternatively, you could take screen captures, recorded in a simple table showing dates, images and a brief statement identifying the stage of your process. You could also capture the process using screencasts. Whatever approach you use, be sure to annotate/discuss the changes you have made and why.
- Provide evidence of comprehensively testing your final outcome.
- Address any relevant implications such as usability, functionality, legal/ethical requirements.
- Synthesise the information from the planning, testing and trialling of components to develop a high-quality outcome.
- Discuss how this information lead to the development of a high-quality outcome.
  - This should include evidence of how the outcome addresses relevant implications.

Testing and trialling can be demonstrated by making a brief screencast showing the outcome being comprehensively tested. You can also take screenshots of your screencast and annotate them. This is often easier than trying to screenshot whilst testing where it is easy to ‘forget’ to screenshot a key part of the test. If you prefer, you are also welcome to talk through your testing and submit a brief screencast to support your evidence.

## Assessment schedule/Mahere Aromatawai: Digital Technologies & Hangarau Matihiko 91906 – Developing for Purpose

| Evidence/Judgements for Achievement/Paetae | Evidence/Judgements for Achievement with Merit/Kaiaka | Evidence/Judgements for Achievement with Excellence/Kairangi |
|---|---|---|
| Use complex programming techniques to develop a computer program.<br><br>The student has:<br>● written code for a program that performs a specified task<br>● used complex techniques in a suitable programming language<br>**For example (partial evidence):**<br>The student’s program allows users to enter typical data and outputs on expected cases.<br>Program has a graphical user interface and custom classes (e.g. one class might ‘hold’ the interface and a second class might include help text).<br><br>● set out the program code clearly and documented the program with comments<br>**For example (partial evidence):**<br>Layout is clear, and whitespace has been effectively used.<br>Student has included comments stating what the code does.<br><br>● tested and debugged the program to ensure that it works on a sample set of expected cases<br>**For example (partial evidence):**<br>Student has provided evidence of testing their program. The testing might be missing some of the expected detail. It might miss some testing showing that the program works for unexpected/invalid values.<br>*The examples above are indicative samples only* | Use complex programming techniques to develop an informed computer program.<br><br>The student has:<br>● documented the program with variable/module names and organised comments that describe code function and behaviour<br>● followed conventions for the chosen programming language<br>**For example (partial evidence):**<br>If the student has used Python, class names are in CapWords, variable and function names are lowercase, and classes appear before the main routine. Code has clear commenting throughout that describes function and behaviour. The student has used an automated tool to check that their code follows conventions.<br><br>● comprehensively tested and debugged the program in an organised way to ensure that it works on a sample of both expected and relevant boundary cases<br>**For example (partial evidence):**<br>Student provides evidence of systematically testing their final outcome to confirm that it works for expected, and relevant boundary values.<br><br>*The examples above are indicative samples only* | Use complex programming techniques to develop a refined computer program.<br><br>The student has:<br>● ensured that the program is a well-structured, logical response to the task<br>**For example (partial evidence):**<br>Program code is easy to read/understand and has been set up in a logical fashion. Functions and classes have been used to keep distinct tasks separate, and to avoid duplicate code. The program explicitly passes data between classes and functions and avoids the use of global variables. Where the program uses a GUI, the GUI and the underlying code are kept separate, and communicate via a well-defined interface.<br><br>● made the program flexible and robust<br>● comprehensively tested and debugged the program<br>**For example (partial evidence):**<br>Student provides evidence of comprehensively testing their program to show that it works correctly for expected, unexpected and boundary values. It has been structured so that making changes to it is easy. For example, the code uses named constants, clearly defined in a ‘constants’ area. The code uses derived values, such as the length of a list, in place of literals.<br><br>*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.

## Assessment schedule/Mahere Aromatawai: Digital Technologies/Hangarau Matihiko 91907 – Developing for Purpose

| **Evidence/Judgements for Achievement/Paetae** | **Evidence/Judgements for Achievement with Merit/Kaiaka** | **Evidence/Judgements for Achievement with Excellence/Kairangi** |
|---|---|---|
| Use complex processes to develop a digital technologies outcome.<br><br>The student has:<br>• used recognised and appropriate project management tools and techniques to plan and manage the development of a digital technologies outcome<br>• decomposed the digital technologies outcome into smaller components<br>• trialled components of the outcome<br>**For example (partial evidence):**<br>The student has used appropriate project management tools and techniques, such as Agile with a Trello or Kanban board, to plan and manage the development of their outcome. The outcome has been broken down into smaller components and these have been trialled and tested. The components have then been combined into a working outcome.<br><br>• tested that the digital technologies outcome functions as intended<br>**For example (partial evidence):**<br>The student has provided evidence of testing the outcome to ensure that it functions as intended: The testing might be missing some of the desired detail and the program might not work as intended for unexpected data, but works for intended input.<br><br>• addressed relevant implications | Use complex processes to develop an informed digital technologies outcome.<br><br>The student has:<br>• effectively used project management tools and techniques to manage development, feedback and/or collaborative processes<br>**For example (partial evidence):**<br>The student provides evidence of versioning the outcome where new versions either have improved functionality or added features.<br>The student provides evidence of sharing documents/data and managing feedback. For example, they used Google Team Drive to share design ideas and to seek feedback.<br>The student provides evidence of how they managed their work flow. This could include screen captures of a Trello board or any other valid planning tool.<br>The student has managed their versions effectively through the use of structured file and folder naming conventions and back-ups.<br><br>• effectively trialled multiple components and/or techniques<br>**For example (partial evidence):**<br>They have trialled several ways of presenting the selection of program choices via the GUI (multiple components) and selected the one that had the best usability and prevented the user from entering incorrect data.<br>Evidence can be seen in their project management tool(s) and/or logs that includes annotations of the | Use complex processes to develop a refined digital technologies outcome.<br><br>The student has:<br>• synthesised information from the planning, testing and trialling of components<br>**For example (partial evidence):**<br>The student outcome is of high quality as a result of student effectively using project management tools and techniques, to efficiently manage the trialling, testing and refinement process. They have incorporated user suggestions and feedback to improve the usability, aesthetics, and functionality of the outcome.<br><br>The student has:<br>• discussed how the information led to the development of a high-quality digital technologies outcome<br>**For example (partial evidence):**<br>The student has reflected on their use of processes to develop their outcome and provided evidence of how the process helped them to effectively test and trial various components to refine and enhance the design and functionality their outcome.<br><br>*The examples above are indicative samples only* |

| | | |
|---|---|---|
| **For example (partial evidence):**<br>The student outcome is easy to use, fully functional, aesthetically pleasing and honours copyright legal obligations.<br><br>*The examples above are indicative samples only* | component(s) and/or techniques trialled with annotations regarding the outcome or next steps.<br><br>• effectively used information from testing and trialling to improve the functionality of the digital technologies outcome<br>**For example (partial evidence):**<br>The student provides evidence of testing and trialling during development and they have indicated how they improved their outcome using this approach. Evidence can be seen in their project management tools and/or logs that includes annotations of changes made to improve the functionality.<br><br>*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.