**Internal Assessment**

**Achievement Standard:** AS 91896 Use advanced programming techniques to develop a computer program.

***(Version 2)* Level 2 Credits 6**

**Learner’s Name: Lee Lin 11DNG**

**Due Date: Term 3, Day 9, Week 2 (24 July 2025)**

**Instructions**

* Read all instructions carefully.
* Complete **ALL** tasks.
* Ensure your name is on every page of your assessment.
* Check that this assessment has pages 1-7 are the correct order and that none of these pages are blank.

**Conditions**

* The timeframe for completion will be set by your teacher according to a particular classroom situation.
* You must follow legal ethical and moral responsibilities when creating your documents.

**Note to Assessors:**

You will need to provide candidates with a copy of the documented policies and procedures or commonly accepted practices of your organisation, prior to being assessed against this achievement standard.

**Authenticity statement**:

The work that I have submitted for this assessment is my own.

I understand any material used and/or quoted that is not my own must be acknowledged in the appropriate manner. I understand that the use of chatbots, generative AI, paraphrasing tools, or other tools that can automatically generate content is not permitted. Material generated by these tools has not been included in this submission.

I understand any material used and/or quoted that is not my own must be acknowledged in the appropriate manner.

I have acknowledged all direct quotes and references used in this assessment.

I understand that plagiarism and/or collusion with other students may result in disciplinary action and a Not Achieved grade being given.

If submitting a digital copy of this assessment I agree to the above.

I give permission for my assessment to be given to the external assessor for marking.

Student Signature:

04/04/2025m

Before you start this assessment familiarize yourself with the NZQA standard. By doing this this you will be able to ensure you get the best grade possible.

|  |  |  |
| --- | --- | --- |
| **Achievement** | **Achievement  with Merit** | **Achievement  with Excellence** |
| Develop a computer program involves | Develop an informed computer program involves: | Develop a refined computer program involves: |
| Writing code for a program that performs a specified task | Following common conventions for the chosen programming language | Ensuring that the program is a well-structured, logical response to the specified task |
| Using advanced techniques in a suitable programming language | Making the program flexible and robust |
| Setting out the program code clearly and documenting the program with comments | Documenting the program with appropriate names and comments that describe code function and behaviour |
| Testing and debugging the program to ensure that it works on a sample of expected cases | Testing and debugging the program effectively to ensure that it works on a sample of both expected cases and relevant boundary cases | Comprehensively testing and debugging the program |

# Additional Requirements

NZQA also sets a number of requirements.

|  |
| --- |
| The programming language chosen must support the required data types, control structures, advanced programming techniques and have good commenting facilities. |
| The computer program must use:   * 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, sensors, or other external source * produces output |
| use **two** or more advanced programming techniques   * modifying data stored in collections (e.g. lists, arrays, dictionaries) * storing multidimensional data in collections * creating methods, functions, or procedures that use parameters and/or return values * responding to events generated by a graphical user interface (GUI) * using non-trivial string manipulation * using additional non-core libraries |
| Example of ways of making a program flexible and robust include:   * 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. |

**Make sure you are familiar with the NZQA requirements. Ultimately these are the criteria that you will be marked against the requirements.**

**Assessment Task**

Julie runs a party hire store and has a range of items for hire. She needs to keep a track of items that are currently out.

She needs to keep the following details:

* Customer full name
* Receipt number
* The item that is hired
* How many of the item the customer has hired

To ensure good data collection the following are required:

* Customer full name required
* Receipt number any number
* The item that is hired required
* How many of the item the customer has hired between 1 and 500

When an item is returned, she needs to be able to delete it, so it no longer shows.

**Planning**

Before you start writing the program consider what the program will look like and plan how it will be structured.

How will your program present the questions and receive user input?

How will you store the data?

* What variables will you require and what type of data will your variables store (e.g. text, numeric, string, integer, Boolean)? Will you store the 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/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. Wherever possible you should try to ensure that your code has a flexible structure to allow for continued development.

**Testing:**

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 noted own 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. You should ensure that your code is robust and that it handles expected, boundary and invalid cases.

**Commenting:**

Ensure that you comment your code appropriately as you develop it and use variable/module

names and comments that describe code function and behaviour.

**Naming Conventions:**

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.

**These are the requirements you must consider:**

**Requirement 1**

* Your program must be in GUI format using tkinter.
* Your program must work without errors.
* Your must create objects and to use data file handling

**Requirement 2**

* Your program must include ALL the following.
  + Use variables storing at least two types of data (e.g. Numeric, text, Boolean)
  + Uses sequence, selection and iteration control structures (e.g. if and while)
  + Takes input from a user, sensors, or other external source
  + Produces output

**Requirement 3**

* The program must use a minimum of TWO of the following
  + Modifying data stored in collections (e.g. two lists or two arrays or two dictionaries)
  + Storing multidimensional data in collections
  + Creating methods, functions, or procedures that use parameters and/or return values
  + Responding to events generated by a graphical user interface (GUI)
  + Using non-trivial string manipulation
  + Using additional non-core libraries (minimum of 2)
* It is recommended you use two lists (or multidimensional data) and a GUI for this program

**Requirement 4**

* The program must be laid out in a sensible manner

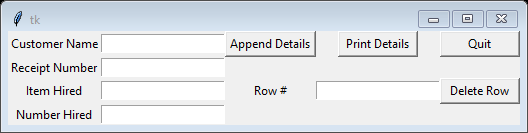
**Requirement 5**

* There must be comments in the code that document what it does

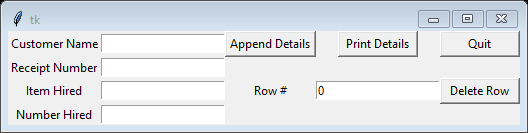
**Requirement 6**

* Provide written evidence that you have tested the program works with a range of expected inputs that get expected results

**Below is a very poor example of what one of the outputs might look like. It might JUST get Achieved**









**The following are also required for you to gain Merit**

**Requirement 7**

* Follow common conventions of the language. These include
  + Variable naming
  + Using separate subroutines for separate tasks
  + Use of constants

**Requirement 8**

* Have comments that fully describe what each step of the code is doing

**Requirement 9**

* Provide evidence that you have tested and debugged the program for a sample of
  + Boundary cases such as ensuring the number of items is between 1 and 500
  + A sample of expected inputs are not left blank

**Requirement 10**

* Provide written evidence that you have tested the program in an organised way – such as using a testing table
* Given written evidence that you have found and fixed errors – such as a written diary or log of changes
* Tested that the program works properly with boundary variables
* Tested that only expected variables are accepted

**The following are also required for you to gain Excellence**

**Requirement 11**

* The program must be well structured and logical in its approach

**Requirement 12**

* It must be flexible and robust. Examples of this include
  + 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.

**Requirement 13**

* It must be thoroughly tested with evidence for all possibilities
* There needs to be comprehensive evidence of debugging such as a diary

**Program checklist**

Ensure you have all these included in your program

* Entry for customer name, receipt number, item name, number of items
* Storing details in a list or lists
* Ability to display the details in a GUI or print
* Ability to delete an item from the list
* Include comments
* Written evidence of testing
* Following naming conventions

**Submission Requirements**

* Signatures from the teacher that they have tested the program at Achieved, Merit or Excellence
* Printed program code
* Printed output demonstrating the program working as expected
* Evidence of testing
* Evidence of debugging (for Excellence)
* A copy of the code must be handed in **Stuwrite/2PAD/91896 Assessment Folder.**

**NOTE:**

Before handing in self-mark your assessment and see if you have handed in everything that is required for this standard. Ensure that the program works on more than one computer.

**GRADE: N / A / M / E**

**Assessment schedule/Mahere Aromatawai: Digital Technologies & Hangarau Matihiko 91896 – Use advanced programming techniques to develop a computer 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.** | **Use advanced programming techniques to develop an informed computer program.** | **Use advanced programming techniques to develop a refined computer program.** | | | |
| **Written code for a program that performs a specified task:**  ☐ Writing a working program to a specified task  ☐ The student has used Python to complete the project  ☐ Program generates a random receipt number | **Has at least one of the following advanced techniques:**  ☐ Program has a working image  ☐ Program generates a random receipt number and places it in a row | **Has included at least ONE of the following:**  ☐ Have a unique receipt number generator  ☐ Have enhanced stylistic features including images and/or icons | | | |
| **Comments**  ☐ The code is internally documented with comments  **Mandatory:**  ☐ Use variables storing at least two types of data (e.g. Numeric, text, Boolean, object)  ☐ Used a GUI library Tkinter  ☐ Use sequence, selection, and iteration control structures  ☐ Take input from a user save to a data file  **The program must have at least TWO of the following advanced techniques:**  ☐ Creating methods, functions, or procedures that use parameters and/or return values.  ☐ Responding to events generated by a GUI  ☐ Using of concatenation  ☐ Using additional non-core libraries  ☐ Have a main()  **Making the program flexible and robust, such as…**  ☐ Creating reusable functions that are found throughout the program  ☐ Conditions ☐ Control structures ☐ Methods  ☐ Has used two functions  ☐ Using constants, variables and derived values in place of literals  **Testing inputs with expected and actual outputs:**  ☐ Every item must be tested, and testing table is provided  ☐ Program works on expected input  *Program may crash on boundary or invalid input.*  ☐ Student has tested all expected cases to qualify the different conditions  ☐ A paragraph is provided explaining the testing process  **Debugging**  ☐ Fixed the program to ensure that it works on a sample set of expected cases. *Students have placed snips/copies of their fixes* | **Comments**  ☐ Every block of code has relevant/descriptive comments  ☐ Comments: Use a space after the #, first word capitalised, use a full sentence ending in a period, indent to the same level as the code it’s describing.  **Naming Conventions**  ☐ The names of the functions and variables are relevant and descriptive  ☐ The name of the variables and functions are in lower case  **Testing valid and invalid inputs with expected:**  ☐ Student has included a test table with all boundaries tested and outputs  ☐ Debugged the program in an organised way to ensure that it works on a sample of both expected and relevant boundary cases (e.g. Max/Min testing)  ☐ On unexpected inputs should generate an error message. Might not be as clear. | ☐ The program is a well-structured, easy to read, logical, detailed and understandable response to the task and inputs  ☐ Functions have been used to keep distinct tasks separate, and to avoid duplicate code  **GUI and the underlying code are kept separate and communicate via a well-defined interface:**  ☐ No console output  ☐ Use of a combo box/spin box  ☐ Formatted and clear entry boxes  ☐ Formatted and clear labels  ☐ Separate message boxes/or very clear instructional messages for valid and invalid inputs  ☐ Made the program flexible and robust – multiple use of methods, all conditions, boundaries  ☐ Program Reads/Writes to a curated .txt, .json, .xls file.  **Comprehensively tested and fixed the program:**  ☐ Program must be bug free  ☐ Every item needs to be comprehensively tested (snips are included)  **Comprehensively testing all:**  ☐ Unexpected inputs  ☐ Expected inputs  ☐ Boundary values  ☐ Detailed and instructional error message for alphanumeric and unexpected inputs (e.g. full stops, symbols, special characters, decimals, ASCII characters)  ☐ It has been structured so that making changes to it is easy. | | | |
|  | **GRADE** | **N** | **A** | **M** | **E** |

**Comments:**

***Final grades will be decided using professional judgment based on a holistic examination of the evidence provided against the criteria in the Achievement Standard***