Te Hoe Rorohiko
Department of Computing

Assessment for AS91637
Version 3
Develop a complex computer program for a specific task

Course Code 91637

Semester Two 2017

Due date: TBA

Credits: 6

Student Name/ID ....

Date	√	Result	Tutor Signature
		Achieved (A) for satisfactory performance	Comment:
		Merit (M) for very good performance	
		Excellence (E) for outstanding performance	
		Not achieved (N) if student does not meet the criteria of the standard	
Entered to Database:			
Date	Signed		

Student has complied with occupational health and safety guidelines and recommendations in relation to working environment and work practices and the requirements of the Health and Safety in Employment Act 1992 and its subsequent amendments.

This assessment has five (5) pages including the cover sheet.

Assessment for: AS91637– 3.46

Construct an advanced computer program for a specified task

Version 3, Level 3, 6 Credits INTERNAL

Achievement Criteria

Achievement ☐	**Achievement with Merit** ☐	**Achievement with Excellence** ☐
Develop a complex computer program for a specified task.	Skilfully develop a complex computer program for a specified task.	Efficiently develop a complex computer program for a specified task.
• Designing and implementing a program that includes variables, an indexed data structure, and a modular structure including details of the procedural structures of the modules	• Following a disciplined design and implementation process, with documented cycles of incremental development	• Ensuring that the overall modular and procedural design, graphical user interface, and event handling design, are a well-structured, logical decomposition of the task
• Including a working graphical user interface with different sources of event generating components and event handling	• Documenting the program with variable and module names and comments that accurately describe code function and behaviour.	
• Using classes and objects to encapsulate data and methods		
• Setting out the program code clearly and documenting the program with comments		• Using variables, constants, and derived values effectively so as to increase the flexibility and robustness of the program
• Testing and debugging the program to ensure it works on a sample of expected input cases.	• Using a comprehensive testing process, to ensure that the program works on inputs that include both expected and boundary cases.	• Comprehensively testing and debugging the program in an organised and time effective way to ensure the program is correct on expected, boundary and invalid input cases.

- The programming language for this standard must be a text-based Object-oriented programming language that supports graphical user interfaces (GUIs) and event based programming.
- The platform for the program could be a personal computer, a mobile device, or a web browser. The program must involve graphical elements such as widgets, images, and/or shapes, and must respond to event based input from the user.
- A complex computer program is one that has a modular structure; an indexed data structure (e.g. array or list); input and output; procedural structures that combine sequential, conditional, and iterative structures; a graphical user interface and event handling; and that includes classes and objects. Inheritance is not required.
- A specified task refers to a set task which requires the development of a complex computer program to resolve. The task must be of sufficient rigour to allow the student to meet the standard and needs to be agreed prior to the program being developed.

INSTRUCTIONS

- Read all tasks carefully.
- Complete **ALL** tasks.

CONDITIONS

- The timeframe for completion will be set by your Assessor according to a particular classroom situation.
- You must follow legal ethical and moral responsibilities when creating your documents.
- You may open and use any previous flowcharts and programs you have created, notes that you have written yourself or the teacher has given you, the Internet and reference manuals.
- Your teacher can give you only very limited guidance, such as telling you where to look if you get stuck.
- You can discuss your ideas with other students but do not show them your plans or program.
- You <u>may not</u> copy others' work. This includes code snippets and flowchart design. All work submitted MUST be your own.

LEARNER'S STATEMENT OF AUTHENTICITY

- I give permission for my assessment to moderated internally, by other schools, or by NZQA.
- This assessment has been completed entirely by me.

Student Name____

Signed ____

Date _________________________

ASSESSMENT SCENARIO

A new company called ‘Flow Computing’ specialises in writing small programs that can be given away to students to help them with their learning.

They want your help to create a program that will help students with their Maths. You will be one of a number of people who will be creating a program, so they want you to come up with your own design without being too limited by design constraints. You are therefore to design your own program. It **MUST** include a Graphical User Interface (GUI). It can be aimed at **ANY** age.

They do however have a few specifications that need to be followed.
(Make sure you read all the specifications before you begin).

TASKS:

1. Plan the program

Create your plan and show it to your teacher before you do any coding. If you later revise the plan, keep the original and submit both for marking.

1.1 The plan should include a flowchart to show the structure of the program.
- It should show the modular structure of the program, including the modules’ names.
- Your flowchart should show how your program will use classes and objects.
- It needs to show where the GUI(s) are called.
- How events (such as clicking and entering text) are handled must be shown.

1.2 The plan also needs a variable table that shows the name, type, and scope of all variables.
- This should include objects, classes and lists.
- Be sure to give all variables appropriate names.
- Give a short description of what each variable is for.

1.3 Create a testing table that can be used when the program has been written.
- It should test for expected variables and boundary variables.
- It should also include unexpected variables.
    - **Warning:** If boundary and unexpected variables cannot be input into your program you will be limiting yourself to a highest grade of Achieved.

1.4 Produce a sketch of how the program will look once written.
- You must hand draw or use a graphics program to produce your sketch.
- Indicate on it the function for each aspect of the design

2. Produce the program

Do not start writing the program until your teacher has approved your plan.

Your plan should be robust enough to allow you to produce your program. If, however, changes are required, update your plan and ensure you keep your original plan as well.

2.1 Ensure your program includes the following:
- Variables that are well named and have suitable attributes.
- Use of classes and objects.
- An array or list - this will most likely be associated with classes and objects.
- Modules (subroutines) that are well structured.
- A Graphical User Interface for the main part of the program.

- Multiple event handling on the GUI – such as mouse clicks or text input boxes.
- Both input and output (for Achieved the Input does not have to be on the GUI).
- Designed to be expanded or changed with relative ease (for Excellence).
- Conditional statements such as IF.
- Use of loops such as WHILE.

3. Document cycles of development

3.1 Keep a record of your development process.

- Keep a diary of what you did and when.
- When plans are changed, produce updated plans, ensuring you keep the originals.
- Detail how you tested each module while the program is being developed.

4. Document the code and lay out the program

4.1 Your code must be documented with comments that explain what each section does.

- A section is a code snippet that does one task. Many sections will make a module.

4.2 Your code needs to be laid out in a sensible and easy to understand way.

- This includes making good use of tabs and blank lines.

5. Test and debug the program

5.1 Test the completed program using your testing table to check it works properly.

- Test for expected, boundary and unexpected variables.
    - Remember that if your program cannot accept boundary and unexpected variables you will limit yourself to a lower grade.
- Make any changes needed to the program (and possibly the plan) to ensure it runs properly.