Achieved	
Criteria	**Requirement at this Level**
Designing and implementing a program	
☐ Includes variables	Variables must be used
☐ Has an indexed data structure	A list or array must be used
☐ Has a modular structure	At least one subroutine/module must be used.
☐ Details the procedural structures of the modules	A flowchart or diagram or text must explain/show the function of the modules
Graphical user interface	
☐ A working graphical user interface is included	A GUI must be included
☐ With different sources of event generating components	At least 2 from widgets, images and/or shapes that the user interacts with
☐ With different sources of event handling	At least 2 from click, enter text, drag etc. with the widgets/images/shapes
Classes and objects	
☐ Using classes and objects to encapsulate data and methods	Classes/objects are required. There is no need to be able to update the objects
Setting out the program code	
☐ Setting out the program code clearly	The code must be understandable and laid out clearly
☐ Documenting the program with comments	Comments must be present. A minimum of what each module is doing
Testing and debugging	
☐ To ensure it works on a sample of expected input cases	Expected values must be tested and the program modified where needed
To develop a complex program	
☐ Text-based event based Object-oriented language is used	No drag and drop languages (such as Alice) used

☐ There must be input and output	Ideally on the GUI, but being on the GUI is not a requirement
☐ Both conditional and iterative must be included	IF and a LOOP must be included. The loop must always run during normal operation
☐ A usable maths program produced	The completed program must teach some aspect of maths

Merit	
Criteria	**Requirement at this Level**
Disciplined designing	
☐ A disciplined design approach is followed	Planning done first including aspects such as flowchart, variable table, sketch(es)
☐ A disciplined implementation approach is followed	Program is developed in a logical and efficient way
☐ Documented cycles of incremental development	Such as revising plan, incremental testing or other suitable evidence
☐ Testing documented	Evidence is provided of testing methods and modifications made
Well-chosen structure	
☐ Well-chosen modules	The program is divided into a number of suitable modules
☐ Well-chosen procedural structure	Within each module the procedures have a sensible structure (most of the time)
☐ Well-chosen scope to variables	Variables are correctly defined as global, local, string, integer etc.
☐ Well-chosen scope and encapsulation for data & methods	Class/objects used properly i.e. created in separate module with its own list/array
☐ Well-chosen graphical user interface	Is a sensible and usable interface that helps to teach some aspect of maths
☐ Well-chosen event handling mechanisms	Clicking, entering data etc. is sensible and a reasonable way to do the task
Naming and documenting	
☐ Suitable variable and module names	The names show what the variable/module is for, and where necessary what it is

☐ Descriptive comments	Comments should accurately describe code function and behaviour
Testing and debugging	
☐ Comprehensive testing	A comprehensive range of combinations of expected variables are tested
☐ Program includes boundary inputs	There must be the ability for boundary inputs to be made
☐ Boundary cases are tested	A range of boundary variables are tested and any problems fixed

Excellence	
Criteria	**Requirement at this Level**
Overall design	
☐ Modules and procedures design is well structured	All module and procedures must be well structured and in a logical order
☐ The graphical user interface design is well structured	The GUI must be laid out in a sensible and logical way
☐ The event handling design is well structured	All the event handling (clicking, text boxes etc.) must be well laid out and work properly
☐ The program is flexible and robust	Any changes or updates must be easy to make and the program is well written
Naming and documenting	
☐ Setting out the program code concisely	The layout of the code is efficient and concise
Testing and debugging	
☐ Time effective testing and debugging the program	The method for testing (such as a testing chart) and actual testing must be efficient
☐ Invalid input cases tested for and handled	Invalid inputs are tested for and responded to in an appropriate way