**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.

**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.