 

**Windows Quiz App**

**Digital Technology NCEA Level 1 2021**

ACHIEVEMENT STANDARD: 91883 1.7v1

Level: 1 Number of Credits: 4 Internal

**Develop a computer program**

ACHIEVEMENT STANDARD: 91884 1.8v1

Level: 1 Number of Credits: 6 Internal

**Use basic iterative processes to develop a digital outcome**

**Overview**

The focus of this assessment is on the design and implementation of a Windows console application that will help a chosen audience to learn about a specified topic.

**Introduction**

This assessment activity requires you to develop a basic quiz that can be used with a chosen audience. For example, you could create a quiz aimed at intermediate mathematics students which allows them to practice calculating the area / perimeter of basic shapes.

You will be assessed on how well you accurately develop your basic computer program through the application of appropriate tools, techniques and user interface. In developing your basic computer program, you must demonstrate evidence of basic iterative processes to plan, trial, and test your outcome.

Your final digital outcome must demonstrate evidence of testing to improve its quality and address social and end-user considerations. For example, you should ensure that your questions are at a suitable level and that the content you choose is appropriate for the chosen audience. It is also important to ensure that copyright is honoured.

You will have approximately 3 weeks to iteratively plan, develop, trial, and test your program. It is **due on the 17th of September 2021**.

To be assessed against these standards, you will need to gather a portfolio of your work from class assignments, homework tasks, and similar evidence. You have had a walkthrough of this during the ‘Lucky Unicorn’ lessons in OneNote so you should have a good idea of what is required.

You may work individually, but if you decide to work together (no more than 2) then you will each be required to provide documentation which shows how the tasks are broken down between the two of you (who is responsible for each component), how you will test your code independently, and how you will then combine the code to test the full program. Moreover, you must each show how you have met the standards individually. One possible way to do this is to have many levels with different types of quizzes at each level, but a common theme across levels. Each person could then create a level. In this case, you would need to discuss and collaborate on common variables, collections of data and/or functions and clearly show which team member created which part of the program.

The following instructions provide you with a way to structure your work to demonstrate what you have learnt to allow you to achieve success in this standard. You must show how you have met all aspects of the standards you are being assessed against including how you have followed legal, ethical and moral responsibilities during development as required for your outcome.

**Version Control**

All students will need to keep several versions of their planning and coding (push to GitHub regularly and/or when you add a new component). If you only submit a final version of your planning, and of your program then it will be very difficult for your teacher to ensure authenticity as code for almost any program can be found on the internet.

To show optimal working you should break the problem into parts and complete each part independently. Keep the code and planning that you do for each part to show authenticity and efficient working.

**Task**

This activity requires you to create a quiz on a topic of your choice. You must decide on the purpose for your quiz. For example, a quiz to help the user to study the road code or a study aid for an upcoming exam in another subject or a quiz to help your classmates learn Te Reo. The quiz should ask the user for their age or year level to determine that the quiz is suitable for them to use.

Decide on the style of quiz that you will make, keeping in mind the time you have available, your programming ability and the software and hardware resources available to you. Examples of quiz styles are listed below:

* A multiple choice quiz
* A word answer quiz
* A timed quiz
* A multi-player quiz
* A game format quiz
* Any other style of quiz.

Your quiz must allow for user input in some form.

**You need to think about:**

* *How your quiz begins.* For example, should you have many levels? Is your quiz only appropriate for a certain age range?
* *How to store your data*. It is advisable to use data stored in collections (e.g lists, arrays or dictionaries) as this will lead to more elegant code. Similarly, the use of user defined functions may help improve the structure, flexibility and robustness of your program.
* *How to structure your program.* What procedural structure will your program require? Will you create functions/method/procedures to improve the structure, flexibility and robustness of your program?
* *How you will give feedback to the user.* This could be in the form of a text prompt, a correct answer should the user answer incorrectly, a score that tracks user progress or any other method that you see fit to use. Also consider if you want to give immediate feedback, or only after quiz is complete.

**You must:**

* Undertake your planning in a manner that suits the outcome and could include sketches, wireframes, storyboards, or mock-ups. You may use online interactive or collaborative planning tools.
* Trial the components, tools and techniques to be included and refine your outcome based on evidence of your trialling. You must show evidence that you have trialled multiple components and/or techniques and have selected the ones that will work best for the purpose of the outcome. Your trialling should be done in an on-going, iterative manner.
* Create this quiz in an iterative manner. For example, break down the program into smaller achievable parts then code, test, debug and evaluate each part before moving on the next progressive improvement. It is advisable to save each version of the code as you make progress.
* Describe a range of social implications and end-user considerations that are relevant to your outcome. Include evidence of how you have addressed these in the process of developing the outcome. For example:
  + How have you addressed ethical or intellectual property issues?
  + How have you ensured that your outcome is usable and functional for your end users?
  + How have you ensured that your aesthetic elements are appropriate for your end users and have enhanced usability?

**Note:** 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, what happens when lives run out or when a sprite gets to the border of the window. It is often useful to note down what you want to test and what you expect to happen, as well as what actually happened.

* Ensure that you comment your code appropriately as you develop it and use variable names and comments that describe code function and behaviour.
* Ensure that you have followed conventions for the programming language of your choice and that you have chosen a well-structured, logical response to the task.
* You should ensure that your code is robust and that it handles expected, boundary and invalid values.
* Wherever possible you should try to ensure that your code has a flexible structure to allow for continued development.

**Note:** Your quiz program must include variables of at least two different types, and sequence, selection, and iteration control structures. The program must also include one or more of: data stored in collections (e.g. lists, arrays, dictionaries); user-defined methods, functions or procedures.

**Hand In:**

Submit all documentation via GitHub. Include your initial planning as well as all the different versions until you have a final version. Submit your first attempt at writing code (which may be for only part of the task or a simplified components of the task). Then also submit subsequent versions of code which show your changes and comments justifying why you changed it. Change the filename or add a new GitHub branch to show each version of the code (i.e. v1, v2, v3, etc.)

**Achievement Criteria:**

**AS91883 (1.7)**

**Develop a computer program**

| **Achievement** | **Achievement with Merit** | **Achievement with Excellence** |
| --- | --- | --- |
| * Construct a computer program. | * Develop an informed computer program. | * Develop a refined computer program. |

**AS91884 (1.8)**

**Use basic iterative processes to develop a digital outcome**

| **Achievement** | **Achievement with Merit** | **Achievement with Excellence** |
| --- | --- | --- |
| * Use basic iterative processes to develop a digital outcome. | * Use basic iterative processes to develop an informed digital outcome. | * Use basic iterative processes to develop a refined digital outcome. |

|  |  |
| --- | --- |
| Description assessment | Term 2/3 project |
| Assessment conditions | Work must be original. Can be done during some classes and at home. |
| Due date | Term 3 Friday 17th September 2021 at 15:00. |
| Further assessment opportunities | None |

Assessment Schedule: AS 91883 Windows Quiz App

|  |  |  |
| --- | --- | --- |
| Evidence/Judgements for Achievement | Evidence/Judgements for Achievement with Merit | Evidence/Judgements for Achievement with Excellence |
| The student has constructed a basic computer program.  The student has written a simple, functional quiz program in the language of their choice. The program may not be structured very well.   * writing a basic program that performs the specified task, using a suitable programming language   **For example (partial evidence)**  *The quiz works as expected but they may have used only one function without parameters.*   * setting out the program code clearly and documenting the program with comments   The student has written the code in a procedural manner, including use of loops and conditions. Comments are present but may not be particularly descriptive of frequent.  **For example (partial evidence)**  *eg. ‘ this code creates the quiz loop*   * testing and debugging the program to ensure that it works on a sample of expected use cases.   The student has shown some evidence of expected use cases that were used to test and debug the program to show that the program works when the user inputs data that is expected.  Testing may be trial and error rather than clearly thought out.  **For example (partial evidence)**  *The student has written about some expected use case data and has tested what happens when the program is run. Such testing may be observed by the teacher, presented in table form with minimal notes, or simply explained using examples.*  *The examples above are indicative samples only* | The student has skilfully constructed a basic computer program.  The student has frequent clear comments throughout the code that helps to describe relevant functions or sections of code.  The variable names clearly describe the data they hold.   * documenting the program with variable names and comments that describe code function & behaviour   **For example (partial evidence)**  *eg. ‘ this function tests that the user input is a number. It will continue to ask for input until the input is a number.*  e.g. blnAnswer = IsNumeric(Console.ReadLine())   * following conventions of the chosen programming language   The student has followed most common programming conventions for their chosen language.  **For example (partial evidence)**  *VB.Net files and functions contain a docstring explaining the purpose of the program/function. Constants are ALL\_CAPS with underscores separating the words if required.*  *Variable names use Camel case.*  *Functions appear below the Main sub. Global vars are declared at the top of a class, thus making the program easier to read.*   * testing and debugging the program in an organised way to ensure that it works on a sample of both expected and relevant boundary use cases.   The student tests frequently during development (observed) and the final program works when the user inputs the data that is expected and checks or handles when the data is outside of specific thresholds.  The student may have kept some form of notes showing what was tested and the outcome of that testing.  Test use cases by student include expected and boundary use cases.  **For example (partial evidence)**  *In a situation where a user is asked to “choose the answer 1,2,3 or 4”, the program checks that the input is above 0 and under 5*  *The program might not correctly handle unexpected data and fail when a word is typed in where a number was expected.*  The examples above are indicative samples only | The student has accurately constructed a basic computer program.   * ensuring that the program is a well-structured, logical solution to the task   The student’s final program consists of multiple general purpose functions that are passed parameters when appropriate. The program flow is clear and there is no unnecessary or redundant code. Functions are well named so that they are self-documenting.  *Functions appear below the Main sub. Global vars are declared at the top of a class, thus making the program easier to read.*  Student has used parameters and arguments to pass data into functions and has returned values for functions, using local variables.  **For example (partial evidence)**  *Function to test input is an integer is named IsInteger()*  *The code is clean, concise, efficient and easily readable. The main program may be short and consists of multiple reusable user defined functions which do most of the logic and processing.*   * making the program flexible and robust   The student has created a program with a flexible structure that can be easily improved upon.  The program uses modules, actions and control structures effectively and without unnecessary repetition. The program uses constants, variables and derived values in place of literals. Such variables appear at the top of the code where they can easily be seen and changed should those values change over time.  **For example (partial evidence)**  *The student has named SCREEN\_NUM as a constant values rather than using 5 (or whatever value a correct answer rates).*   * comprehensively testing and debugging the program in an organised and time-effective way to ensure the program is correct on expected, boundary and invalid use cases.   The student makes sure the program checks the validity of input data and deals with expected, boundary and unexpected or invalid data. All use cases have been tested including testing each condition within their code.  **For example (partial evidence)**  *Testing has been done in a systematic way. Test cases have been well-thought out and notes may have been made showing that the code works as expected for all use cases. They have used “try- except” error handling to ensure that program handles even unexpected events or inputs.*  *The examples above are indicative samples only* |

Assessment Schedule: AS 91884 Windows Quiz App

|  |  |  |
| --- | --- | --- |
| Evidence/Judgements for Achievement | Evidence/Judgements for Achievement with Merit | Evidence/Judgements for Achievement with Excellence |
| Use basic iterative processes to plan and develop a digital outcome.  The student:   * Plans a digital outcome to address a problem, need, opportunity or interest.   **For example (partial evidence):**  *The student researches issues relating to quizzes, examines user preferences and plans how they are going to incorporate these through creating sketches and interactive wireframes. The student has used an online tool to plan out the development process.*   * Develops the digital outcome by decomposing the problem   **For example (partial evidence):**  *The student decomposes their basic computer game into the components that need to be developed and tested such as graphics, functions, user interface, etc.*   * Plans and trials components of the outcome in an iterative manner.   **For example (partial evidence):**  *The student plans and tests the code for the quiz loop. They next plan and test the question and answer part of the quiz. They next plan and test the feedback part of the quiz. The check that quiz is easy to use and understand. Each component is planned and tested in an iterative manner until the final game quiz is produced.*   * Tests that the programming outcome functions as intended   **For example (partial evidence):**  *The student plans testing the functionality of the quiz with various users to ensure the game works.*   * Describes the social implications and end-user considerations that are relevant to the outcome.   **For example (partial evidence):**  *The student spoke to students from their target audience to determine their quiz preferences. The student recognises that it is unethical to use copyrighted questions. They have recognised that quiz layout will affect the enjoyment of using the quiz. However, the student may not have chosen the best solution to address the considerations or could have more fully addressed these considerations.*  *The examples above are indicative samples only* | Use basic iterative processes to plan and develop an informed digital outcome.  The student:   * Uses information from testing and trialling to improve the outcome.   **For example (partial evidence):**  *The student provides screen shots with a brief annotation that shows the improvements in the quiz mechanics that were made after making changes. They also provide a short video to demonstrate improved functionality after correcting a bug in the code.*   * Trials multiple components and/or techniques and selects those which ensure the outcome functions as intended.   **For example (partial evidence):**  *The student trials two different techniques for performing a particular aspect of their quiz and selects the choice that does not cause functionality issues. The student trials two different question generating techniques and chooses the one that is most efficient.*   * Addresses the social implications and end-user considerations that are relevant to the outcome.   **For example (partial evidence):**  *The student addresses that fact that it is unethical to use copyrighted questions by making up their own questions / using creative commons material that has been correctly attributed. They also ensure that any images used in the quiz are copyright free. They have addressed usability and aesthetic considerations through testing their game with a range of end users.*  *The examples above are indicative samples only* | Use basic iterative processes to plan and develop a refined digital outcome.  The student:   * Synthesises information from planning, testing and trialling of components to develop a high quality outcome.   **For example (partial evidence):**  *The student has provided evidence that their planning has allowed them to meet project timelines and include all the planned for components and information. Their outcome functions as intended and has no obvious errors in functionality or presentation of the information. Evidence gained from trialling and thorough and organised testing has been integrated into the outcome in an on-going manner to ensure the outcome is of high quality, including aesthetics, functionality and usability.*  *The examples above are indicative samples only* |