|  |  |
| --- | --- |
| **Achievement** | ✔✘ |
| **Criteria** | **Tick** |
| **Use complex programming techniques to develop a computer program** |  |
| writing code for a program that performs a specified task |  |
| using complex techniques in a suitable programming language (eg C# using classes and files) |  |
| setting out the program code clearly and documenting the program with comments |  |
| testing and debugging the program to ensure that it works on a sample of expected cases. |  |
| **Use complex processes to develop a digital technologies outcome** |  |
| using recognised and appropriate project management tools and techniques to plan the development of a digital technologies outcome (eg GitHub and Trello) |  |
| decomposing the digital technologies outcome into smaller components |  |
| trialling components of the outcome |  |
| testing that the digital technologies outcome functions as intended |  |
| addressing relevant implications. |  |

|  |  |
| --- | --- |
| **Merit** | ✔✘ |
| **Criteria** | **Tick** |
| **Use complex programming techniques to develop a computer program** |  |
| documenting the program with appropriate variable/module names and organised comments that describe code function and behaviour |  |
| following conventions for the chosen programming language |  |
| testing and debugging the program in an organised way to ensure that it works on a sample of both expected cases and relevant boundary cases. |  |
| **Use complex processes to develop a digital technologies outcome** |  |
| effectively using project management tools and techniques to manage development, feedback and/or collaborative processes |  |
| effectively trialling multiple components and/or techniques  eg trialling multiple ways to develop components |  |
| effectively using information from testing and trialling to improve the functionality of the digital technologies outcome. |  |

|  |  |
| --- | --- |
| **Excellence** | ✔✘ |
| **Criteria** | **Tick** |
| **Use complex programming techniques to develop a computer program** |  |
| ensuring that the program is a well-structured, logical response to the task |  |
| making the program flexible and robust |  |
| comprehensively testing and debugging the program. |  |
| **Use complex processes to develop a digital technologies outcome** |  |
| synthesising information gained from the planning, testing and trialling of components |  |
| discussing how this information led to the development of a high-quality digital technologies outcome. |  |

|  |  |
| --- | --- |
| 1. The programming language chosen must support the required data types, control structures, complex programming techniques, and have good commenting facilities. 2. *A complex computer program*:  * uses variables storing at least two types of data (e.g. numeric, text, Boolean, object) * uses sequence, selection and iteration control structures * takes input from a user, file, sensors, or other external source * produces output * uses two or more complex programming techniques.  1. Examples of *complex programming techniques* include (at least 2 elements):  * programming or writing code for a graphical user interface (GUI) * reading from, or writing to, files or other persistent storage * object-oriented programming using class(es) and objects defined by the student * using types defined by the student * using third party or non-core API, library or framework * using complex data structures (e.g. stacks, queues, trees).  1. 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. | 1. Examples of *project management tools and techniques* include:  * Agile or waterfall techniques * Kanban or scrum boards * version control software * collaboration tools * managing assets.  1. Examples of *relevant implications* include:  * social * cultural * legal * ethical * intellectual property * privacy * accessibility * usability * functionality * aesthetics * sustainability and future proofing * end-user requirements * health and safety. |