| **Number** | **Version** | **Title** | **Credits** | **Assessment** |
| --- | --- | --- | --- | --- |
| AS91906 | 1 | Use complex programming techniques to develop a computer program | 6 | Internal |

|  |
| --- |
| **Submission Checklist (Achieved)** |

To meet the achievement level criteria it is important that your code meet the minimum expectations of the assignment. Before submitting, please complete the checklist below to ensure you have not missed anything significant in your submission.

|  |  |
| --- | --- |
| Checklist | Done? Y/N |
| My program uses variables storing at least two types of data (e.g. numeric, text, Boolean) | Y |
| My program uses sequence, selection (IF) and iteration (LOOP) control structures | Y |
| My program uses Input from a user, sensors or another external source & produced output | Y |
| My program uses two or more complex programming techniques. | Y |
| My code is set out clearly using suitable whitespace | Y |
| I have included comments to document and explain what the code is doing | Y |
| I have prepared a video recording of my program working and will submit it with this document. | N |
| I have completed the appropriate testing table(s) to show how effective my program is. | N |
| I have copied my completed code into the space provided in this document | Y |
| I have prepared a video recording of my program working and will submit it with this document. | N |

|  |
| --- |
| **About Your Project** |

In the space below, tell me anything about your program that I may need to know.

Include elements such as…

* What features did you add beyond the basic brief?
* Were there any changes to the brief that were agreed by your teacher?
* What environment is needed for your code to run?
* Etc.

In short, explain anything beyond hitting run button for a standard python environment and expecting the program to act as the example program provided.

|  |
| --- |
|  |

|  |
| --- |
| **1. Achieved Level Documentation** |

# 1.1 - 1.6

Evidenced within the code

# 1.7 Complex Tools

In the table below identify the complex tools you have used, why they have been used and where the evidence of their use can be found.

|  |  |  |
| --- | --- | --- |
| Complex Tool | Why was it used | Where is it used |
| *e.g. Student created classes/objects* | *To generate new ….* | *These are created …* |
| Classes/objects |  |  |
|  |  |  |
|  |  |  |

|  |
| --- |
| **2.Merit Level Documentation** |

# 2.1 – Variable Names & Comments

Evidenced within code

# 2.2 - Conventions Used

Each programming language has a set of conventions that should be followed. For Python the current conventions are documented in the [PEP 8](https://www.python.org/dev/peps/pep-0008/) style guide and contain guidance such as:

* Variable and function names should all be lower case with words separated by underscores.
* Lines of code should not be longer than 79 characters
* Lines of comments should not be longer than 72 characters
* Functions should always contain a docstring

Other languages such as [C#](https://docs.godotengine.org/en/3.1/getting_started/scripting/c_sharp/c_sharp_style_guide.html) and [JavaScript](https://www.w3schools.com/js/js_conventions.asp) also have their own style guides.

In the table below identify the conventions you have followed and provide evidence that they have been applied.

|  |  |
| --- | --- |
| Selected Programming Language | Style Guide Used |
| C++ | Google style guide |
| Evidence that Conventions have been applied | |
| *e.g.1 - A screen shot of the output from an online checking tool like* [*https://www.codewof.co.nz/*](https://www.codewof.co.nz/)  *e.g. 2 - A description of all of the conventions you have taken with examples.* | |
|  | |

**NOTE: This section is intended to demonstrate that you are aware of and have adhered to the conventions for your selected language. A screen shot alone does not demonstrate that you have been successful throughout your entire project so make sure that you thoroughly check each aspect of your work.**

|  |
| --- |
| **3.Excellence Level Documentation** |

In the table below explain how you have addressed each of the refined characteristics of the assessment and where this can be seen in your program.

|  |  |  |
| --- | --- | --- |
| Refinements | Explanation | Where is it seen |
| 3.1 The program code is efficiently and effectively organised |  |  |
| 3.2 The program has features to help manage user input |  |  |
| 3.3 The program has user friendly output |  |  |
| 3.4 The program includes error capture features |  |  |
| 3.5 The program report unexpected behaviour to the user |  |  |

|  |
| --- |
| **4.Final Submission & Testing** |

# 4.1 - Your Video

Record a video showing your game in operation with valid inputs. If you wish to show any error capturing you have included in your program, record these in a separate video.

Be sure to take your time and give the viewer of this video and opportunity to fully see your game in action. Save your video recording in a suitable file format (e.g. wmv or mp4 – NOT an iSpring file)

|  |
| --- |
| Below tell us the name of this file and where it is stored. |
|  |

# 4.2 – Demonstrate Authenticity

Evidenced by Teacher throughout project development & after submission

# 4.3 Valid “Expected” Input Testing (Achieved)

Valid “Expected” tests are tests that show your program operates as expected if the inputs received are as you expect them to be. For each input test that your program works using valid inputs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test No. | Test (include test data if necessary) | Expected Result | Actual Result | Test Result |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |
| 11 |  |  |  |  |
| 12 |  |  |  |  |
| 13 |  |  |  |  |
| 14 |  |  |  |  |
| Etc. | Please add more tests as needed |  |  |  |

# 4.4 Boundary Input Testing (Merit)

Boundary tests are tests that show your program operates as expected if the inputs received are at the extreme ends of the allowed input range. It is also worth including the other side of each boundary to show that the boundaries are actually operating correctly. E.g. To test an input asking for a number between 1 and 10, apply 4 tests: 1 then 10, but also 0 and 11!

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test No. | Test (include test data if necessary) | Expected Result | Actual Result | Test Result |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |
| 11 |  |  |  |  |
| 12 |  |  |  |  |
| 13 |  |  |  |  |
| 14 |  |  |  |  |
| Etc. | Please add more tests as needed |  |  |  |

# 4.5 Comprehensive Invalid Input Testing (Excellence)

**Comprehensive testing needs to show testing of all areas of the program.**

Invalid tests are tests that ensure the program operates as expected regardless of the inputs made by the user. This will be the biggest of all of the testing sections.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test No. | Test (include test data if necessary) | Expected Result | Actual Result | Test Result |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |
| 11 |  |  |  |  |
| 12 |  |  |  |  |
| 13 |  |  |  |  |
| 14 |  |  |  |  |
| 15 |  |  |  |  |
| 16 |  |  |  |  |
| 17 |  |  |  |  |
| 18 |  |  |  |  |
| 19 |  |  |  |  |
| 20 |  |  |  |  |
| Etc. | Please add more tests as needed |  |  |  |

# Your Code

Please copy the final code from your program into the space below.

|  |
| --- |
| Final Code |
| <allcode.txt> |