Computer Program/Code/Software Marking Criteria

Your program/code/software will be assessed against the following criteria:

* **Readability**: *The ease with which your code/software can be understood by a human. Usually helped by: an abundance of clear, concise, informative comments; the use of naming conventions; a consistent and standard source code layout (often achieved via appropriate code indentation/whitespace).*
* **Structure**: *The degree to which the code has been organised into relevant blocks, files and other structures as appropriate.*
* **Validity:** *The absence of logical or functional errors and the presence of appropriate tests and internal checks.*
* **Efficiency**: *The extent to which the code reduces unnecessary computation and limits memory usage.*
* **Functionality**: *The overall ability of the software to solve a specified problem or help the user achieve their ends, including the usability and aesthetics of Graphical User Interfaces and websites.*

“Ancillary documentation” in the table below is any essay about the development process and/or diagrams or user guides that you are asked for in addition to the code. You will not always have been asked for this. Where you have not been made aware of terms (for example, “loose coupling”), either in lectures or further reading, you will not be expected to match these standards, though you are, of course, welcome to. What the criteria mean will also change as you learn, so, for example, early on “good structure” may mean code that is clearly organised within a single block, while later it may mean placing code into procedures or classes, once you know those terms. It should be noted that the code/software not compiling or running does not *necessarily* mean the work will be classed as a fail, especially if an earlier version of the code/software does work, but there is a failure that has occurred as a result of attempting to add some extra functionality. The final mark may result from a combination of elements from across the marking scheme.

|  |  |
| --- | --- |
| **Grade (UG / TPG)** | **Typical Criteria** |
| **Super First / High Distinction** | **Readability**: The code/software will have excellent documentation including (if/as appropriate) ancillary documentation which may detail a professional [development process](https://en.wikipedia.org/wiki/Software_development_process). The documentation may include: [structural diagrams](https://en.wikipedia.org/wiki/Unified_Modeling_Language) that visualise how the code is structured; data flow or workflow diagrams that visualise how a data flows through the software in a processing workflow. The software source code will contain appropriate useful comments that explain what the code does. The code will be laid out clearly and consistently adhering to a style guide. The program itself will come with structured help files which may be embedded throughout the system.  **Structure**: Where appropriate, the code will follow standards of [loose coupling](https://en.wikipedia.org/wiki/Coupling_%28computer_programming%29) and [high cohesion](https://en.wikipedia.org/wiki/Cohesion_%28computer_science%29), and will utilise professional [design patterns](https://en.wikipedia.org/wiki/Software_design_pattern). Associated Graphical User Interfaces and websites will be well structured, in a manner that potentially shows the influenced of [Information Architecture](https://en.wikipedia.org/wiki/Information_architecture) literature. For websites this will include well designed structuring of associated directories and files, with an exemplary separation of functionality (HTML; CSS; JS).  **Validity**: The code will show an appropriate level of testing and internal checks, both to ensure the program’s validity and to prevent fatal issues during running.  **Efficiency**: The code will be close to optimal efficiency in terms of both memory and processor usage.  **Functionality**: The code will include the core functionality, enhanced with additional functionality if this is appropriate for the task set. There will be checks to help prevent the system exiting in error (especially as a result of user providing unusual input values or clicking buttons and pressing keys wildly). The software will be designed to produce informative error messages that advise a user if an error is encountered. The system will be designed to prevent users making mistakes and/or that mistakes are resolved with minimum of difficulty. There will, where appropriate, a well-designed set of [usability](https://en.wikipedia.org/wiki/Usability) tests detailed in any ancillary documentation. Any associated Graphical User Interfaces or websites will be well presented and consider user accessibility issues and any write up will consider [Human-Computer Interaction](https://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction) literature.  Overall, the code will be of a professional release standard, or close to it. |
| **First / Distinction** | **Readability**: The code/software will have excellent documentation including (if/as appropriate) ancillary documentation which may detail a professional [development process](https://en.wikipedia.org/wiki/Software_development_process). The documentation may include: [structural diagrams](https://en.wikipedia.org/wiki/Unified_Modeling_Language) that visualise how the code is structured; data flow or workflow diagrams that visualise how a data flows through the software in a processing workflow. The software source code will contain appropriate useful comments that explain what the code does. The code will be laid out clearly and consistently adhering to a style guide. The program itself will come with structured help files which may be embedded throughout the system.  **Structure**: Where appropriate, the code will show some thought into the development of structural units (like classes) that enhance the reusability and readability of code. Associated Graphical User Interfaces and websites will be thoughtfully structured. For websites this will include good structuring of associated directories and files, with separation of functionality (HTML; CSS; JS).  **Validity**: The code will show some evidence of additional checking, especially to prevent errors arising during the running of the program.  **Efficiency**: The code will show that some thought has been put into efficiency and there will be some evidence that the code has been tested to ensure it is sufficiently efficient. Functionality: The code will include the core functionality, enhanced with additional functionality where appropriate. An attempt will have been made reduce the number or handle user induced errors. There will, where appropriate, some discussion of [usability](https://en.wikipedia.org/wiki/Usability) in ancillary documentation, and usability will be a core consideration in any associated Graphical User Interfaces and websites.  Overall, the code will be of a near professional release standard with only minor usability issues. |
| **IIi / Merit** | **Readability**: The code/software will have appropriate documentation, including (if/as appropriate) ancillary documentation which may detail the thought processes behind the code functionality and development. The software source code will contain appropriate useful comments that explain what the code does. The source code will be laid out with only minor inconsistencies.  **Structure**: Where appropriate, the code will show some thought into the development of structural units (like classes) resulting in clear code, but perhaps not really facilitating code reuse. Associated Graphical User Interfaces and websites will show thought in their structuring, with appropriate separation of functionality (HTML; CSS; JS).  **Validity**: The code will include standard elements of checking and elements to prevent errors.  **Efficiency**: The code may be inefficient and ancillary documentation may not provide any evidence that work has been done to make the code efficient.  **Functionality**: The code will include the core functionality, with only minor issues. Associated Graphical User Interfaces and websites will work and provide access to the core functionality. |
| **IIii / Pass** | **Readability**: The code/software will have limited documentation. There may be little evidence that thought has gone into code structuring or standards. There may only be minimal inline comments. There will be an attempt to lay out the code neatly, but there may be some inconsistency. Any asked for ancillary documentation may be of limited detail.  **Structure**: The code may only follow broad structures associated with the coding languages or suggested as starting points for the assessment. Where there is an expectation of code separation, much of the code may nevertheless be in a single file or block. Graphical User Interfaces and websites may not function well or there may not be a clear separation of functionality (HTML; CSS; JS).  **Validity**: Minor errors may be encountered at runtime or during compilation.  **Efficiency**: The code may be inefficient.  **Functionality**: There will be evidence of an attempt to include the core functionality, although minor issues may have been encountered. Key elements of Graphical User Interfaces and websites are expected to work, but these may have multiple issues. |
| **III / Fail for Masters** | **Readability**: The code/software will have little or no documentation. Inline comments will be minimal or non-existent, or may contain errors or misunderstandings. The code layout may be somewhat confusing and difficult to read. Any ancillary documentation may be of very limited or use. Issues encountered in developing the code/software are not likely to be well explained.  **Structure**: The code will not be structured and may, for example, be in a single block where this is inappropriate for the language in question. For Graphical User Interfaces and websites, key links between different sets of code or functionalities may be broken.  **Validity**: The code may fail to compile or during runtime it may not run as expected due to functionality being incorrectly implemented.  **Efficiency**: The code may be very inefficient.  **Functionality**: There may be evidence of an attempt to include the core functionality, but this will not work. Graphical User Interfaces and websites may be confusing, distracting, and not adhere to standards. |
| **Fail for Undergraduate and Major Fail for Masters** | **Readability**: The code/software will have no documentation. Inline comments will be absent or may be misleading and reveal significant misunderstanding. Layout will be confusing and the code will be difficult to read.  **Structure**: The code will be unstructured or structured badly. For Graphical User Interfaces and websites, key links will probably be broken.  **Validity**: The code may fail to compile or during runtime it may not run as expected due to functionality being incorrectly implemented.  **Efficiency**: The code will be very inefficient.  **Functionality**: The code will fail to implement nearly all of the core functionality, and/or will do a completely different job, and/or be a random collection of code fragments. Graphical User Interfaces and websites may be completely broken and confusing. |

Computer Code Mark and Feedback Sheet

For information about the criteria code is marked against, see the associated *Computer Code Marking Criteria*.

General Class Comments

Individual Feedback

*In general, your overall mark will reflect the average of the categories below, however please note that the grades for the individual components are nevertheless indicative only and the overall mark may vary from this. For example, a lecturer may decide that a very nice but minor piece of code warrants a higher mark in total, despite not making a huge difference to any one of the individual categories. Please read the detailed feedback in order to understand your mark.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Undergrad → | Super first | First | IIi | IIii | III | Fail |
| Masters     → | (High) Dist | Distinction | Merit | Pass | Fail | Fail |
| Readability |  |  |  |  |  |  |
| Structure |  |  |  |  |  |  |
| Validity |  |  |  |  |  |  |
| Efficiency |  |  |  |  |  |  |
| Functionality |  |  |  |  |  |  |
| Overall |  |  |  |  |  |  |

Feedback

Areas to especially prioritise

|  |  |
| --- | --- |
| Marker | Provisional mark |
|  |  |

*Any mark given here is provisional and subject to moderation by the School’s Board of Examiners. This is to ensure comparable marking standards for all students. In a minority of individual cases moderation can lead to either the raising or lowering of the provisional marks.*