## Project rubric for (authors' names):

| Criterion | Expert (1) | Skilled (2) | Apprentice (3) | Learner (4) | (5) |
|---|---|---|---|---|---|
| **Appearance** | ***Information clearly, attractively presented.*** Fonts reflect importance of items. Graphics clarify topics. Structure helps audience to understand. | Information clear and attractive. Font or graphics confusing. Structure helps audience to understand. | Information presented clearly. Font uneven. Structure and graphics are confusing or unclear. | Structure, graphics and text unclear or missing. Fonts uneven. | |
| **Scientific language** | ***Authors' use of technical language shows they understand their subject's background knowledge.*** They minimise their use of jargon and define all necessary terms and acronyms. | Language shows authors understand the background knowledge of the subject. Jargon is appropriate. Not all terms and acronyms are defined. | Language shows that authors understand the background knowledge of the subject. Too much jargon and undefined acronyms. | Informal language: authors do not seem to understand the subject properly. Some scientific terms incorrectly. | |
| **Code structure** | ***Code is clearly organised and formatted***: short, ***coherent*** methods, ***logical indentation***, and clear linebreaks (lines under 100 characters) make the code easy to follow. | Code is easy to read with minor formatting/indentation mistakes, e.g.: bracket-matching. | Code is generally easy to follow, but logical formatting is poor. | Code is readable only by someone who knows what it is supposed to be doing. | |
| **Clarity and coherence** | ***Code follows a clear, consistent conceptual metaphor.*** Program header clearly states this metaphor. Coding components (comments, variable and method names) *consistently* declare their role by reference to metaphor. | Conceptual metaphor is present, but unclearly stated. Coding components mostly refer to this metaphor for clarity. | Conceptual metaphor is present, but unclearly stated. Relationship of coding components to this metaphor are generally unclear. Use of magic numbers. | Program has no clear conceptual metaphor. | |
| **Comments** | ***Comments indicate clearly what code is doing***, using clear, simple language that is appropriately positioned and formatted. | Header and inline comments make the code easier to understand. | Inline comments are embedded in the code and separate logical code sections. | Inline comments are embedded in the code. | |
| **Variable naming** | ***Variables' names express simply and briefly their purpose in the program.*** | Variable names are awkwardly long but express their purpose clearly. | Variable names express only unclearly their purpose in program. | Variable names express their purpose only very vaguely. | |
| **Data types** | ***Variable types (array, logical, …) are used efficiently*** to produce correct results. | Variable types used efficiently to produce mostly correct results. | Variable types are used efficiently but produce incorrect results. | Variable types are used inefficiently/incorrectly. | |
| **Control structures** | ***Control structures (selection, iteration, …) are used efficiently*** to produce correct results. | Control structures efficiently used to give mostly correct results. | Control structures used efficiently but produce incorrect results. | Control structures are used inefficiently/incorrectly. | |
| **Modularity (modules, functions)** | ***Modular architecture is clear and easy to follow.*** Data and method responsibility cleanly factorised into modules to minimise rippling. | Modularity is clear and easy to follow but responsibility allocation permits some rippling. | Modularity is easy to follow but global data permit excessive rippling. | Data-, but not method-, responsibility is allocated modularly. | |
| **Validation** | ***Program fulfils all specifications, and performs exception-checking*** for errors and out-of-range data. | Program runs and meets all specifications. Performs some checking for entry and range errors. | Program produces correct results but displays them incorrectly. Some checking for entry and range errors. | Program gives correct results but displays them incorrectly. No error-checking. | |
| **Efficiency** | ***The code is highly efficient***: stores multiply used data, reduces processing steps without sacrificing readability or comprehensibility. | The code is efficient without sacrificing readability or comprehensibility. | The code is fairly efficient without sacrificing readability or comprehensibility. | Code is inefficiently patched together from mismatching partial solutions. | |
| **Total =** | | **/ 11 =** | | | |