Assessment of software design and development with a General Purpose Programming Language

This example rubric defines key criteria for:
- user interface design, and the use of relevant tools such as mockups,
- algorithm design, and the use of relevant tools such as flowcharts and pseudocode,
- the development (coding) of the algorithms in a General Purpose Language, such as Python or JavaScript, including testing.

The criteria in this rubric are not suitable for assessing a complete digital solution. They do not address:
- the overall design thinking process and the value of innovation,
- problem definition and identification of solution requirements,
- evaluation of completed solution,
- collaborative work and project management.

ASSESSMENT STANDARD (extracts - Digital Technologies)

Years 5 and 6: Design solutions by developing algorithms to address the problems. Incorporate decision-making, repetition and user interface design into their designs and implement their digital solutions, including a visual program.

Years 7 and 8: Design user experiences and algorithms incorporating branching and iterations, and test, modify and implement digital solutions.
<table>
<thead>
<tr>
<th>Achievement standard criteria</th>
<th>E</th>
<th>D</th>
<th>C</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
</table>
| **Design user interface**  
(text only) | No demonstrable attention to interface efficiency or effectiveness for the purpose of the solution. | Interface is inefficient and/or ineffective for the purpose of the solution. | Interface is somewhat efficient and effective for the purpose of the solution. | Interface is efficient and effective for the purpose of the solution. | Interface is highly efficient and effective for the purpose of the solution. |
| User is presented with no instructions to interact with the solution. | User is presented with adequate instructions to interact with the solution. | Some responses to invalid user input. | Some useful responses to invalid user input. | User is presented with clear, complete instructions to interact with the solution. | User is presented with clear, complete instructions to interact with the solution. |
| No response to invalid user input. | Little response to invalid user input. | | | Appropriate, helpful responses to invalid user input. | |
| **Design user interface**  
(graphical) | No demonstrable attention to interface efficiency or effectiveness for the purpose of the solution. | Interface is inefficient and/or ineffective for the purpose of the solution. | Interface is somewhat efficient and effective for the purpose of the solution. | Interface is efficient and effective for the purpose of the solution. | Interface is highly efficient and effective for the purpose of the solution. |
| Mockup missing or incoherent. | Mockup inadequate for presenting design. | Mockup used to present design adequately. | Mockup used correctly to present design. | Mockup used correctly to present design thoroughly. | |
| Interface lacks key elements and is inadequate for the intended user. | Elements of the interface are unclear, incomplete and/or unsuitable to the intended user. | Elements of the interface are adequate in terms of clarity and suitability to the intended user. | Most elements of the interface are clear and suitable to the intended user. | All elements of interface are clear, complete and suitable to the intended user. | |
| When present, colour, placement and font choices do not demonstrate understanding of design conventions. | Colour, placement and font choices make the solution adequately usable, and demonstrate basic understanding of design conventions. | Colour, placement and font choices make for a mostly consistent experience, and demonstrate moderate understanding of design conventions. | Colour, placement and font choices make for a consistent experience, and demonstrate strong understanding of design conventions. | | |
| **Design algorithm** | Little or no apparent use of flowchart / pseudocode. | Flowchart / pseudocode used sparingly or incorrectly. | Flowchart / pseudocode used adequately, with some conventions followed. | Flowchart / pseudocode used mostly correctly, with most conventions followed. | Flowchart / pseudocode used correctly, with all necessary conventions followed. |
| Algorithm is incomplete or incoherent, demonstrating little or no understanding of sequence. | Algorithm is inefficient and/or ineffective, demonstrating limited understanding of sequence. | Algorithm is somewhat efficient and effective, demonstrating adequate understanding of sequence. | Algorithm is efficient and effective, demonstrating moderate understanding of sequence. | Algorithm is highly / maximally efficient and effective, demonstrating strong understanding of sequence. | |
| Algorithm does not correctly incorporate more than one or two of the following: | Where appropriate, algorithm correctly incorporates some of the following: | Where appropriate, algorithm correctly incorporates most of the following: | Where appropriate, algorithm correctly incorporates: | Where appropriate, algorithm correctly incorporates: | |
| • iteration (loops)  
• branching (decisions)  
• variables  
• user input  
• output | • iteration (loops)  
• branching (decisions)  
• variables  
• user input  
• output | • iteration (loops)  
• branching (decisions)  
• variables  
• user input  
• output | • iteration (loops)  
• branching (decisions)  
• variables  
• user input  
• output | |
<table>
<thead>
<tr>
<th>Develop code – overall functionality</th>
<th>Basic functional code is missing.</th>
<th>Code is inadequate or contains syntax errors.</th>
<th>Code is adequate but not complete, or contains syntax errors.</th>
<th>Code is largely complete and free of syntax errors.</th>
<th>Code is fully complete and free of syntax errors.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program cannot run.</td>
<td>Program is not functional or contains bugs that affect basic functionality.</td>
<td>Program contains bugs that affect some functionality. It can be run successfully with minor modifications.</td>
<td>Program meets some functional requirements, reflects design somewhat.</td>
<td>Program meets most functional requirements and mostly fulfills design.</td>
<td>Program meets all functional requirements and fulfills design.</td>
</tr>
<tr>
<td>No demonstrable attention given to functional requirements or design.</td>
<td>Program meets little or no functional requirements.</td>
<td>Little or no validation code.</td>
<td>Validation code sometimes in place.</td>
<td>Validation used when necessary, giving some useful feedback to user.</td>
<td>Validation correctly used when necessary, giving useful feedback to user.</td>
</tr>
</tbody>
</table>

| Develop code – programming skills | Few or none of the following skills are utilized correctly:  
- iteration (loops)  
- branching (decisions)  
- variables  
- user input  
- output | Some of the following skills are utilized correctly:  
- iteration (loops)  
- branching (decisions)  
- variables  
- user input  
- output | An adequate range of the following skills are utilized efficiently:  
- iteration (loops)  
- branching (decisions)  
- variables  
- user input  
- output | Most of the following skills are utilized efficiently:  
- iteration (loops)  
- branching (decisions)  
- variables  
- user input  
- output | Where appropriate, all of the following skills are utilized thoroughly and efficiently:  
- iteration (loops)  
- branching (decisions)  
- variables  
- user input  
- output |

<table>
<thead>
<tr>
<th>Develop code – readability &amp; internal documentation</th>
<th>Code is largely incoherent.</th>
<th>Code shows no attention to organisation.</th>
<th>Code shows limited attention to organisation.</th>
<th>Most code organised clearly and logically.</th>
<th>All code organised clearly and logically.</th>
</tr>
</thead>
</table>
| No attention to rules and conventions to maximise code readability. | Little or no appropriate rules and conventions followed to maximise code readability, including:  
- tabbing and whitespace  
- naming of variables / functions | Some appropriate rules and conventions followed to maximise code readability, including:  
- tabbing and whitespace  
- naming of variables / functions | Most appropriate rules and conventions followed to maximise code readability, including:  
- tabbing and whitespace  
- naming of variables / functions | Comments are mostly present where appropriate and are mostly clear. | All comments present where appropriate, thorough and clear. |
| Comments are not present or randomly inserted. | Comments are rare. | Some comments are present. | | | |

| Develop code - testing | No formal testing apparent. | Formal testing sporadic and rare. | Formal testing includes some or one of the following (where appropriate):  
- unexpected user input / data  
- out of range data (boundary checking)  
- wrong type data | Formal testing includes most of the following (where appropriate):  
- unexpected user input / data  
- out of range data (boundary checking)  
- wrong type data | Formal testing includes all of the following (where appropriate):  
- unexpected user input / data  
- out of range data (boundary checking)  
- wrong type data |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No evidence of testing.</td>
<td>Testing tool (eg. testing table) used incorrectly or sparingly.</td>
<td>Testing tool (eg. testing table) partly complete and used adequately.</td>
<td>Testing tool (eg. testing table) mostly complete and used correctly.</td>
<td>Testing tool (eg. testing table) complete and used effectively.</td>
<td></td>
</tr>
</tbody>
</table>