Computer Science – SDLC Moderation Task 2021

# Mental Maths Skills Builder

## The Task

Your task is to create a simple maths game to help someone practice their mental arithmetic skills.

The person playing the game will be able to choose the type of problem (e.g. addition, subtraction, mixed…) *and/or* the level of difficulty (e.g. easy, medium, hard).

The problems will be displayed one at a time, and the user will be able to enter an answer for each problem. The answers will be automatically be graded and the results displayed.

## Stages of the Software Development Life Cycle (SDLC)

Requirements

Design

Implementation

Testing

Release & Maintain

**Deliverables**

1. A Word document (or equivalent) that details the following information for each stage of the Software Development Life Cycle.
2. Source code (i.e. your \*.java files).
3. Screenshots demonstrating your program running.

**Requirements**

* **Objectives** – What are you aiming to produce? Be specific!
* **User Requirements** – What does the user need the software to do? What does the software allow the user to do?
* **Functional Requirements** – What does the project need to be able to do so that the user's requirements are met?

**Design Documentation**

* How will the program do what it needs to do?
* It is important to discuss at least *two alternatives* within this phase, and *give reasons* for final choices
* Design the GUI (Graphical User Interface), including:
  + Layout of components (e.g. drop downs, text fields, buttons, display areas, etc.)
  + Data flow (how data links from one screen to another, or how information links.)
* Use images, dataflow diagrams, to give as much detail in this phase of the documentation so that it is visual and easy to understand for clients

**Implementation - Technical Documentation**

Technical Documentation is documentation written during the implementation phase. It details how the program does what it does.

Included in this phase are:

* Comments in code – essential for self or for others
* *(Optional stretch goal: If a custom class is used, include a UML class diagram)*

**Test – Plan and Report**

Devise a testing plan to ensure you have thoroughly tested all aspects of your program and describe how you have done this.

If your requirements document is clear and comprehensive, you should even be able to specify your testing plan before writing code.

At the very least, create a systematic list of inputs and/or program states and the expected outcomes.

For example…

|  |  |  |
| --- | --- | --- |
| **Test Criteria** | **Date** | **Outcome** |
| Checking for incorrect input   * Negative numbers * Words rather than numbers |  | Allows negative numbers to be put in – this needs updating. |
| Calculations come out correctly |  |  |
| Works on different machines/platforms |  |  |
| Accessibility options |  |  |

**Release and Maintain - User Documentation**

User documentation is any documentation written for the end user, system administrator, or support staff. It typically contains instructions on:

* setting up the program
* how to use the program
* what buttons do what
* how to do basic functions, etc.

# Assessment

**Criterion 6: apply the software development life cycle to a variety of problems**

The learner…

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Rating C** | **Rating B** | **Rating A** |
| **1** | writes straightforward programs which meet specifications using appropriate programming standards and a range of programming constructs | follows the Software Development Life Cycle (SDLC) to write complex programs that meet the specifications using the appropriate programming standards | writes well designed complex programs using the SDLC which meet the specifications using programming standards with an appropriate user interface |
| **2** | accesses and applies core support resources to assist in writing programs | uses appropriate programming constructs and accesses relevant resources | evaluates possible programming constructs and resources, and chooses the most appropriate |
| **3** | tests programs against a plan and assesses how well the programs perform | specifies detailed testing plans before a program is written and makes some program revisions after testing | specifies comprehensive testing plans before a program is written, uses both hand and automated tracing to debug programs and refines programs in response to the testing |
| **4** | correctly follows a given set of design principles for the user interface. | explores options for the user interface regarding the specified problem. | tests and evaluates options for the user interface for a solution and selects the most appropriate. |