|  |  |  |
| --- | --- | --- |
| **Software Implementation Checklist** | **Faculty of Computing, Engineering and the Built Environment** | New Logo Tiny |

|  |  |
| --- | --- |
| Please fill in your name and student ID in the table below. | |
| **Student Name** | *Hani Khaled Hisham Kamal* |
| **Student Number** | 22102516 |
| **Course and Year** | MSc. Computer Science 2021-22 |
| **Module Code** | CMP7244 |
| **Module Title** | Software Development |
| **Module Leader** | Dr Aliyuda Ali |
| **Assessment item:** | 1.2 A game program development in Python |

|  |  |  |
| --- | --- | --- |
| Students are required to complete this software implementation checklist for their Python game. First, you need to tick alongside the name of the game chosen for the assessment 1.2. Then, select only the features that were implemented in your code. You can select all/some features from any marking range as long as they have been implemented in the code submitted for this assessment. For example, you can select all/some features from the marking range 40%-49% and all/some features under the range 70%-79% etc. This implementation checklist should be submitted with the test documentation for the assessment 1.2.  **Important notice**: This checklist will assist the tutors when marking your code, hence, you should only select the feature requirements that are implemented in your code. Even if some features are not working correctly, you can select them as long as there is evidence in your code showing the implementation attempt. However, it is not acceptable for a student to claim the implementation of features that were not attempted/implemented in the game. False claims are a clear indication that the student does not understand the submitted code, hence, the submission will be investigated further for plagiarism, and the tutor marking the assessment may invite the student to explain all/parts of the submitted code. | | |
| 1. **Marking criteria for the implementation of the Minesweeper game** | **x** | |
| **Achieving a mark of 40% to maximum of 49%** | | |
| The game **must** implement **all** the following: | | |
| * One player plays on a single 9 \* 9 grid. | | **x** |
| * The computer should randomly select 10 grids, and secretly insert mines inside these grids. | | **x** |
| * The player should navigate the grid by selecting the grid to reveal its hidden content. | | **x** |
| * If a square contains a mine the player loses the game, and the rest of the grid is revealed showing the locations of the rest of the mines. But, if the grid contains no mine, the grid is tagged with a colour or character to indicate a clear field. This grid cannot be selected again throughout the game. | | **x** |
| * The game should display the number of hidden mines, and count the number of clear grids discovered by the player. | | **x** |
| * The game should announce if the player wins the game that is revealing all clear grids. | | **x** |
| * Text based user interface is used to play the game. | | **x** |
| * Computer player plays randomly. | | **X** |
| **Achieving a mark of 50% to maximum of 59%** | | |
| The game **must** implement **all** the above and the following: | | |
| * Development of Basic Graphical User Interface (GUI) using Python Tkinter | | **x** |
| * The game can be restarted at any stage of the game. | | **x** |
| * While navigating the game, if the selected grid contains no mine, a digit is displayed in the square, indicating how many adjacent squares contain mines. On the other hand, if no mines are adjacent, the square becomes blank, and all adjacent squares will be recursively revealed. | | **x** |
| **Achieving a mark of 60% to maximum of 69%** | | |
| The game **must** implement **all** the above and the following: | | |
| * The game can be saved and loaded into and from a text file. | | **🞎** |
| * To help avoid hitting a mine, the grid of a suspected mine can be marked by flagging it. The grid can also be unmarked. | | **x** |
| * The grid size should be configurable, i.e. the player should be able to select the number of columns, rows and mines before the start of the game. | | **x** |
| * Random computer player should play the game by navigating the grid. | | **x** |
| * Time delay should be introduced between each computer guess to observe computer moves. | | **x** |
| * Time delay should be introduced in-between computer guesses to allow easy observation of the computer moves. | | **x** |
| **Achieving a mark of 70% to maximum of 79%** | | |
| The game **must** implement **all** the above and the following: | | |
| * Computer player should associate random moves with some kind of rules to increase its chances of winning. | | **x** |
| **Achieving a mark of 80% and over** | | |
| The game **must** implement **all** the above and the following: | | |
| * The computer player should have 3 levels of intelligence. | | **x** |
| * + Simple – completely random | | **x** |
| * + Medium - Computer player should associate random moves with some kind of rules to increase its chances of winning. | | **x** |
| * + Advanced – Computer player should implement AI algorithms. For example, using history data and/or scoring system. For example, before the computer player selects a grid, it will look at the digits to determine selection. | | **x** |