

**Programming 1 (PRG1)**

Year 1 (2019/20), Semester 1

**SCHOOL OF INFOCOMM TECHNOLOGY**

Diploma in Financial Informatics

Diploma in Cyber Security & Forensics

Diploma in Information Technology

Diploma in Common ICT Programme

# ASSIGNMENT CHECKLIST

**Due on 5 August 2019 (Monday), 8.30 am**

**Individual/Team/Both:** Individual

**Format:** Completion Statuses

Additional Features

Validation Statuses

Function Descriptions

There are a total of 7 pages (including this page) in this handout.

**Submission: You are to submit this checklist together with the source code for the assignment in a .zip file via MEL.**

|  |
| --- |
| ***WARNING***  ***If a student is found to have submitted work not done by him/her, he/she will not be awarded any marks for this assignment. Disciplinary action will also be taken.***  ***Similar action will be taken for the student who allows other student(s) to copy his/her work.*** |

**1. OBJECTIVE**

This assignment checklist provides the student’s assignment completion statuses of basic and additional features of the assignment.

**2. BACKGROUND**

This assignment checklist is provided to facilitate the tutors’ testing and verification of work done as declared by the student.

**3. SCOPE**

This assignment checklist shall cover all features (both Basic and Advanced requirements) specified in the assignment document.

**4. COMPLETION STATUSES**

The following table shall provide in detail completion statuses for the **Basic** requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| S/NO | Feature | Parts of the feature developed and implemented | Remarks |
| 1. | Display main menu | The function for displaying the main menu and prompting the user has been implemented. User input validation has been put in place (See “validation” below). The main menu will always appear. | N/A |
| 2. | Read and load maze from file | A function for reading and loading a maze from file has been implemented. The user will be prompted for a relative path to the maze layout file. Validation of the maze file is in place (See “validation” below). Maze layouts with big, small or “odd” dimensions will be accepted and will function correctly. | N/A |
| 3. | View maze | A function for viewing the maze works file. If no maze has been loaded since startup, the default 8x8 maze is loaded. The maze layout is shown to the user. | N/A |
| 4. | Play maze game | A function for playing thee maze game is implemented and functions as expected. If no maze layout has been loaded since startup, the default 8x8 maze is loaded. The current maze layout, character position and controls are shown to the user. The user can exit at any point in the game by keying in “M” (not case sensitive). This would lead to the game being reset and no scores being added to the leaderboard. User input validation has been added (See “validation” below). | N/A |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Configure current maze | A function to configure the maze has been implemented where the user can key in the |  |
| 6. | Export maze to file | A function Exporting maze has been implemented and will override any files with the same path. |  |
| 7. | Create new maze | A function for creating a new maze is implemented. The user is prompted the dimensions of the maze (Starting at 2x2). An empty, functional and valid maze with the “A” and “B” positions is generated and set. User input validation is in place. |  |
| 8. | Exit maze | Exiting the maze during console and sensehat play works. The user will be returned to the main menu and the game will reset. No score is added to the leaderboard. |  |

The following table shall provide in detail completion statuses for the **Advanced** requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| S/NO | Feature | Parts of the feature developed and implemented | Remarks |
| 1. | Play maze using SenseHAT | Any 8x8 maze is playable and basic user controls will function as expected. The user may exit the game by pressing down on the joystick. Furthermore, the joystick controls are shown to the user through the console. Any dimensions bigger than 8x8 is not supported. | A screen reset has been implemented. However, the screen does not reset. The maze is laterally inverted. However, basic wall and out-of-bounds detection and controls works as expected. |
| 2. | Validation | All user input is validated and sanitized. The program will not crash due to an unexpected user input. This includes leaderboard.json, and any maze.csv file. Should an unexpected user input be keyed in, the program will feedback to the user and re-prompt the user in most cases. Under certain cases, the program may launch the user back to the main menu. Should leaderboard.json be non-existent or corrupted during startup, the program will prompt the user to choose to either disable the leaderboard for the current session or reset the leaderboard. Should leaderboard.json become corrupted during a session, the program will disable viewing or adding to the leaderboard for the rest of the session.  Should a maze.csv with variable width be loaded, playing with the “console” will work fine. Out-of-bounds detection also works as expected. Should a maze.csv without “A” or “B” positions be loaded, default “A” and/or “B” positions will be set automatically.  Should a leaderboard entry contain information longer than the leaderboard column width, the information will be shortened appropriately (e.g. names will be appended with triple-dot “…”, large scores will be have a plus appended “999999+”)  Should an invalid path be keyed for maze layout import or export, the program will present the user a user-friendly error along with the full error. This is an intentional design to allow the users to know the reason for a failed import or export.  Overall, all errors are properly handled and feedback to the user with ErrorHandler().new\_error(). All of the errors (except those that may occur with Play with Sensehat due to odd dimensions) will occur due to user input error.  KeyboardInterrupt is caught. However, the program will honor the request and exit gracefully. | Validation is made possible through utilization of “try…except” blocks which are then properly handled with (e.g. With ErrorHandler().new\_error()). |
| 3. | View leaderboard | Should there be no entries, a “No leaderboards” message would appear.  Otherwise, the leaderboard would should the maze layout followed by the scores for that maze layout beneath it. All maze layouts and score entries saved to leaderboard.json will appear.  Should leaderboard.json be corrupted after startup but before “view leaderboard” is selected, the “view leaderboard” option will disable itself.  Should a score, time or name be longer than the table’s column width, the information will be shortened automatically.  Should a score be below 0, the score will appear as “< 0”. |  |

**Note:**

* ***You are expected to declare upfront on the actual statuses.***
* ***The functions declared shall be exact to the ones presented at the time of the presentation.***
* ***You are required to show your solution code to your tutor during the presentation. Your tutor will go through solution code with the student to verify and assess your understanding of your work. Your tutor may ask you to implement some change requirements to the assignment.***
* ***NO MARKS will be awarded for the advanced features if all the basic features have NOT been fully implemented (and fully working).***
* ***Marks will be deducted if you are not able to show your understanding of the program, both basic and advanced features (if applicable), during the presentation.***
* ***Additional features delivered should be in alignment to the objective of the original assignment’s intent.***