

**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 | e.g.  - a function for display main menu  - obtain user input  - check user input to go to the respective options  - continue to ask for user input and end the program if exit option is chosen.  A function (Game().\_main\_menu()) has been implemented that…   1. Calls Menu().render() to generate a standardized menu layout in a string 2. Obtains user input with ControlAdapter 3. Keeps prompting user for input if previous input is invalid    1. Catches “non-int” invalid input    2. Catches “out-of-range” invalid input    3. Invalid input is feedback-ed to user using ErrorHandler().new\_error() - A standardized error message generator.   This function is only to be called by Game().start() - Which is the overall class “loop bootstrapper” – Which it itself is to be only called by the root main() function – An overall program “bootstrapper”.  The root main() function is separate as its scope is beyond that of the Game class. | Nil. |
| 2. | Read and load maze from file |  |  |
| 3. | View maze |  |  |
| 4. | Play maze game |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Configure current maze |  |  |
| 6. | Export maze to file |  |  |
| 7. | Create new maze |  |  |
| 8. | Exit maze |  |  |

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 |  |  |
| 2. | Validation |  |  |
| 3. | View leaderboard |  |  |

The following table shall list all the functions delivered:

**Function Descriptions**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S/No | Function name | Description | Input Parameter(s) | Expected output |
|  | main | Bootstraps the entire program by being responsible to call Game().start() | N/A | None |
|  | Game() | An instantiated class that is responsible for showing details to the user, getting the user input and calling the correct functions based on the user input. | N/A | None |
|  | Game().start() | Bootstraps the game logic by acting as a “loop bootstrapper” to keep re-showing the main menu (by calling Game().main\_menu).  This allows the game to “restart” after every “cycle” (after any main menu option is completed) | self | None |
|  | Game().\_main\_menu() | Internal function that renders the main menu and calls the relevant internal function corresponding to the options. | self | None |
|  | Game().\_maze\_configure() | Internal function  Shows possible user configuration opts. and runs either:   1. self.\_maze\_configure\_add\_passageway 2. self.\_maze\_configure\_set\_startpoint 3. self.\_maze\_configure\_set\_endpoint | self | None  OR GameEscapeCodes.main\_menu |
|  | Game().\_maze\_configure\_add\_passageway() | Gets user input for coordinates by calling internal function self.\_maze\_configure\_option\_menu().  Checks if the user input requests to back out to main menu or config menu and honors it.  Calls self.\_maze.set\_block(), passing the coordinates and MazeBlocks.ground (A static string that dictates what “block” to set at those coordinates). | self | None  OR  GameEscapeCodes.main\_menu  OR  GameEscapeCodes.config\_menu |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**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.***