

**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 | | 1. display main menu  2. obtain user input  3. check user input to go to the respective options  4. continue to ask for user input and end the program if exit option is chosen. | menu() is used in this feature | |
| 2. | | Read and load maze from file | | 1. display title  2. obtain file name  3. convert data from file to nested list  4. display number of lines in file  5. return to main menu when file loaded successfully  6. return maze list to main menu  7. user input validation (more info below) | load() is used is used in this feature | |
| 3. | | View maze | | 1. display title  2. return to main menu when press “ENTER”  3. obtain the maze list and print the maze of any size  4. will show error message when no maze is loaded | map\_viewer() and map() is used in this feature | |
| 4. | | Play maze game | | 1.obtain the maze list  2. display layout  3. use a duplicate copy of the maze when playing the game  4. print the maze  5. check for A and B point coordinate and show to user  6. obtain user input  7. when is a move option is chosen:  1. check and stop next move when next move is a wall or will be outside the maze (will show an error message to user when true)  2. reset the current A point location to path  3. set new A point into the maze  8. set maze back to default when exit to main menu  9. show game over message to user when reach B point  10. continue to ask for user input and exit to main menu if exit option is chosen.  11. the number of moves, time start, time end and number of passageways on maze is recorded for leaderboard  12. user input validation (more info below)  13. will show error message when no maze is loaded | | map(), game\_check(), game\_move(), game\_over() and play\_game() is used in this feature |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Configure current maze | 1. obtain the maze list  2. obtain user input for what to configure  3. check user input to go to the respective options  4. continue to ask for user input for what to configure and exit to main menu if exit option is chosen.  5. obtain user input for the coordinate to make the change  6. check user input and make the change  7. can choose to exit to configuration menu or main menu  8. user input validation (more info below)  9. return new maze list to main menu  10. will show error message when no maze is loaded | map(), config\_menu(), config\_option(), config\_maze() is used for this feature |
| 6. | Export maze to file | 1. display title  2. obtain user input for new file name  3. convert current maze list to string than create the file | export() is used for this feature |
| 7. | Create new maze | 1. display title  2. warn user current maze will be remove  3. obtain user input  4. create new maze using the detail from user input | create\_map() is used for this feature |
| 8. | Exit maze | 1.exit the program | nil |

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 | 1.obtain the maze list  2. convert maze list from color code so sensehat can understand  3. display layout  4. use a duplicate copy of the maze when playing the game  5. display the maze to sensehat  6. check for A and B point coordinate  7. obtain joystick event  8. when a joystick (expect middle click) event is detected:  1. check and stop next move when next move is a wall or will be outside the maze  2. reset the current A point location to path  3. set new A point into the maze  9. exit to main menu when joystick middle click is press  10. set maze back to default when exit to main menu  12. show game over message to user when reach B point on sensehat  13. the number of moves, time start, time end and number of passageways on maze is recorded for leaderboard  14. will show error message when no maze is loaded  15. will show error message when no sensehat is detected, and bring user back to main menu | map(), sensehat\_map(), game\_check(), game\_move(), game\_over() and play\_game() is used in this feature |
| 2. | Validation | 1. check for input in main menu, if input is invalid, show an error message to user and ask them to try again  2. check for input in load(), if file cannot find or user input invalid symobl, show an error message to user and bring them back to main menu  3. check for input in play\_game(), if input is invalid, show an error message to user and ask them to try again  4. check for input in config\_menu(), if input is invalid, show an error message to user and ask them to try again  5. check for input in config\_option(), if input is invalid, show an error message to user and ask them to try again  6. check for input in create\_map(), if input is invalid, show an error message to user and ask them to try again |  |
| 3. | View leaderboard | 1. display title  2. get the leaderboard information from leaderboard.csv and put into a list  1. if cannot find leaderboard.csv, it will create a new empty leaderboard.csv  3. display top 10 player with the higher score with the time it takes to complete the game  4. leaderboard is sort using the score in descending order  5. add new score to the leaderboard when game over | leaderboard() is used for this function |

The following table shall list all the functions delivered:

**Function Descriptions**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S/No | Function name | Description | Input Parameter(s) | Expected output |
|  | box\_formatting() | Align center message given | message | output |
|  | gui\_menu() | A custom layout to print message | title, messages, action, need\_input | none |
|  | clear() | Do 40 “\n” to clear the console | N/A | none |
|  | menu() | Print the main menu and obtain user input | N/A | inputs |
|  | maps() | Print the maze | map | none |
|  | load() | Obtain file name from user. Read and format the file information to a list | N/A | maze\_list |
|  | map\_viewer() | Print option 2 title, call maps() and return to main menu when press “ENTER” | maze\_list | none |
|  | game\_check() | Check if item is needed can be found before starting the option | types, maze\_lists, a, b | True or False |
|  | game\_move() | When playing, check the move user input is able to move, if able to, “A” to new location | dlt, dlt\_value, a, row, col, game\_map, move, types | a, game\_map, move |
|  | game\_over() | To inform user that game over, ask for user name for leaderboard and validate the name and send to leaderboard() | scoring | False |
|  | play\_game() | A function that responsible for initialization and a loop that obtain input like move and exit game. This is use for both console and sensehat game. During initialization, it will duplicate the maze list, get the A and B point coordinate and call game\_check() to see is everything is ready before starting the game. When everything is ready, it will set the start time to scoring than start the loop to get user input, if the input is a move it will call game\_move until the gameover than it will call game\_over() or when user want to exit the game | maze\_lists, sensehat, types | none |
|  | config\_menu() | To display configure menu and obtain input on which option to configure, the input will be give to config\_option() | maze\_list | maze\_list |
|  | config\_option() | It will display the maze list and ask user what coordinate to make the change. It obtain the input, validate and format it than pass it to config\_maze() | option, values, types, maze\_list | maze\_list |
|  | config\_maze() | Take the coordinate from config\_option() and make the change to that coordinate to the maze. If it is to set A or B point, it will auto remove the old point to a wall | inputs, values ,types, maze\_list | maze\_list |
|  | export() | Obtain file name from user, than convert maze list to string and create and write the string to the file | maze\_list | none |
|  | create\_map() | To create a empty map from what input obtain by user, a warning will be show to user before overwriting the old maze list | N/A | maze\_list |
|  | leaderboard() | A function that responsible for reading, adding and showing the leaderboard. It will check if Leaderboard.csv exist, if not, it will create one empty Leadboard.csv, than it will convert the detail in the file to a list for easier to read  During adding, it get parameter from game\_over() | type, name, score, time | none |
|  | sensehat\_map() | To convert maze list from nested list to list and “X”, “O”, “A”, “B” to color code | a, game\_map | maps |

**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.***