# CPT113 Hackathon 2

Semester 2, Academic Session: 2022/2023

Prepared by Dr. Teh Je Sen, Dr. Nur Hana Samsudin

| Course Outcomes (CO), Program Outcomes (PO), Taxonomy Level (LT) and Soft Skills (SS) | **At the end of this course, students will be able to:** | **PO** | **LT** | **SS** | **Assessment Method** |
|---|---|---|---|---|---|
| | Use data structures in problem solving and object-oriented programming | PO1 | C3 | | Test (13), Final Exam (PA) |
| | **Construct object-oriented programming with the appropriate data structures.** | **PO2** | **P4** | **CTPS** | **Hackathon (03)** |
| | Analyze the requirements and design for objectoriented programming and problem solving. | PO3 | C4 | CTPS | Test (13), Final Exam (PA) |

## 1 Hackathon Scenario

As a game developer, you are given the task of producing a complete text-based video game. You need to build the game entirely in C++. The game can be of any genre, ranging from role-playing to multiplayer card games. Examples of text-based games include Zork and The Dark Room.

Apart from being elaborate enough to show the level of understanding of class and multiple class processing with relationships, your program **must** include the use of **linked lists** or any of its variations like **stacks** or **queues**. These ADTs must play a **core role** in your game, such that the game will not be able to function efficiently without the linked list. For example, a linked list could be used to save the game's state, such that a player can choose to load the game from a previous state.

## 2 Program Specifications

Here are some basic requirements for the hackathon:

- The game must be fully functional and can be compiled with any IDE or just a Linux terminal using GCC or G++. You are responsible for ensuring cross-compatibility.
- Program must be extensive enough to showcase a high level of understanding of classes and multiple class relationships.
- Program must include the use of a linked list, stack, **OR** queue ADT as one of its core components. The ADT must play an important role in the game, and not be used just to store simple information.
- Classes consist of relevant attributes and proper processing.
- The program must maintain data encapsulation of the classes and respect the concept of information hiding amongst classes.

- You must use multiple file inclusion to have a more readable program.
- The `main()` doubles as the main menu of the game, and allows the user to start a new game, choose the number of players, load their previous game, or any other required functionalities.

Your program **must** have the following:

- **Dynamic** linked list, stack, or queue ADT.
- The game should create and store two or more nodes in the linked list.
- A text file must be used to store historical data and must be reused when the program is restarted. The program will read from the text file if it exists (the text file acts as a simple database).
- Input validation
- Gameplay – easy to follow, intuitive
- Good interface design and flow
- Meaningful comments in the source codes
- Coherent gameplay or game storyline
- Nice to have features: Multiple players, save/load game feature

All codes and documentation must be uploaded to a GitHub repository. Ensure that this repository is private until the submission date. After the submission date, the GitHub repository must be made public, and no further changes are allowed.

**Important Note 1:** Your program complexity and clarity will also contribute towards your marks.

**Important Note 2:** You will be required to record a video to showcase and demo your game. The demonstration of your game must match the actual game uploaded to GitHub. Note that no modifications to your code are allowed after the due date. Any updated codes on GitHub after the due date will incur a penalty.

**Important Note 3:** It will be a plus if your game is fun to play!

## 3 Documentation Format and Guidelines

There is no report. Instead, you are required to upload all of your source codes to a GitHub repository. You will create a ReadMe.md file in your GitHub repository to document your game. Here are some example repositories for your reference:

- `https://github.com/itajaja/hb`
- `https://github.com/BKcore/HexGL`
- `https://github.com/opentomb/OpenTomb`

To format your ReadMe file, you can refer to this link: `https://tinyurl.com/github-format`. Your ReadMe.md file and repository should contain the following:

- Description of your game.
- Features of your game.
- How to play your game.
- How object-oriented concepts were used to develop your game.
- How linked lists/stacks/queues play a role in your game.
- Screenshots of your game.
- A link to your game demo video.
- Your codes (**only .h and .cpp files**). Do **NOT** upload project files from your IDE. Anyone should be able to compile your code using other IDEs, GCC or G++.

## 4 Other Restrictions

- You must **NOT** use global variables (unless it is a type **const**).
- You must **NOT** use vector, list, queue, or any possible data structure provided by the built-in C++ library.
- You must **NOT** use <vector>, <list>, <linkedlist>, <queue>, <stack>, or any other preprocessor directives that have never been used during classes and tutorials. You may use only preprocessor directives that you have learned during your class and lab sessions.
- You **MUST** use multiple file inclusion.
- No public member variables.
- You must **NOT** update your codes after the deadline or face a penalty.
- Your GitHub repository should be made private until the hackathon deadline.

**Gameplay Video:** Rather than a presentation, you are required to create a gameplay video or demo for your game. The video should highlight the features of your game, how to play your game, and how it works. The gameplay video **must** match the actual user experience when we try your game. Clearly highlight any bugs or incomplete features if any. The link to the video must be included in your GitHub ReadMe file.