CS-2001 Data Structures (FALL-2023)

Semester Project Statement: Design and Implement a Race Car Game

Project Description:

In this semester project, you will have the opportunity to apply your knowledge of data structures in a practical and exciting way by designing and implementing a race car game in C++. The goal of the project is to create an engaging and interactive game that incorporates various data structures, such as graphs, linked lists, or queues, to manage game elements efficiently.

Project Objectives:

1. Develop a 2D console-based race car game using C++ without utilizing any external graphics library. Implement a text-based user interface within the console (A text-based user interface (TUI) relies on characters and text to convey information to the user instead of graphical elements. In a console-based game, a TUI might involve using ASCII characters and text to represent various game elements). The game must also feature player-controlled race cars, obstacles, tracks, and a scoring system. Here's a simple example of what a text-based user interface for a race car game might look like:

2. **Data Structures Integration:**

a. Graphs for Map and Navigation:

- Map Representation: Create a graph to represent the game map. Nodes in the graph represent locations or waypoints in the game world, while edges represent paths between them. This graph can be used for player navigation.
- Navigation and Pathfinding: Implement algorithms like Dijkstra's algorithm or any other shortest path finding algorithms on the graph to find optimal paths for player-controlled race cars. The graph can help ensure that characters navigate the race track efficiently and avoiding obstacles

b. Queues for Obstacle:

- Obstacle : Utilize a queue data structure to manage the generation of obstacles. New obstacles can be added to the queue as they are generated, and the game loop can process the queue to introduce these elements at appropriate times.
- Example: If your game features randomly appearing obstacles on the race track, you can enqueue obstacle objects with information about their type and location. During each game update, you can dequeue these objects and add them to the game world at the specified positions.

c. Linked Lists for Collected Coins and Game Trophies:

- Collected Items: Maintain a linked list to keep track of items collected by players during the game, such as coins or trophies. Each node in the linked list represents an item collected, with relevant information (e.g., item type, score value).
- Example: As players collect coins or achieve milestones, you can add nodes to the linked list. The linked list allows easy tracking of collected items and can be used to update the player's score or display their achievements.

3. Game Logic:

Develop the game logic, including player controls, collision detection, scoring, and win/lose conditions. Implement gameplay features that challenge players and make the game enjoyable.

4. User Interface:
Design and create a user-friendly interface (text-based user interface TUI) for the game, including menus, scoreboards, and any additional features that enhance the player experience.

5. Documentation:
Provide clear and comprehensive documentation for your code, including comments, code organization, and explanations of data structure usage.

6. Testing and Optimization:
Thoroughly test the game to ensure it runs smoothly and without bugs. Optimize the code and data structure implementations for performance and efficiency.

Instructions (before starting the Project):
1. Project to be done in a group of 2.

2. Apply all validations for invalid inputs.

3. Plagiarism: Plagiarism of any kind (copying from others, copying from the internet, etc.) is not allowed. If found plagiarized, you will be awarded zero marks in the project.

Submission Guidelines

a. submit both .h and .cpp file Your submission must contain your name, student-id, and on the top of the file in the comments. Example the first line of project you should write //Maheen_Arshad_22i111. Missing this will result in 20% marks deduction.

b. Move your .cpp and .h file in one folder. The folder must contain only submission.cpp and h files(no binaries, no exe files etc.,). If we are unable to download your submission due to any reason you will be awarded zero mark.

c. Run and test your program on machine before submission. If there is a syntax error, zero marks will be awarded in that specific question.

d. Rename the folder as ROLL-NUM_SECTION (e.g. 21i-0001_A) and compress the folder as a **zip file**. (e.g. 21i-0001_A.zip). Only **zip file** will be acceptable.

e. Submit the .zip file on Google Classroom within the deadline. Late submission will be marked zero. No exceptions

f. Submission other than Google classroom (e.g. email etc.) will not be accepted.

g. The student is solely responsible to check the final zip files for issues like corrupt
files, viruses in the file, mistakenly exe sent. If we cannot download the file from Google classroom due to any reason it will lead to zero marks in the project.

h. A detailed project report documenting the design and implementation process, including data structure usage.

Grading Criteria:

Your project will be assessed based on the following criteria:
1. Functionality: Does the game work as intended? Is it enjoyable and engaging for players?
2. Data Structure Usage: How effectively and appropriately did you use data structures (graphs, linked lists, or queues) to improve the game's performance and functionality?
3. Code Quality: Is the code well-organized, well-documented, and free of errors or memory leaks?
4. User Interface: Is the user interface intuitive and visually appealing?