DS Project Documentation

SUBMITTED BY:

Muhammad Usman Toseef 22I-2562

Mohammad Hassaan Ejaz 22I-2434

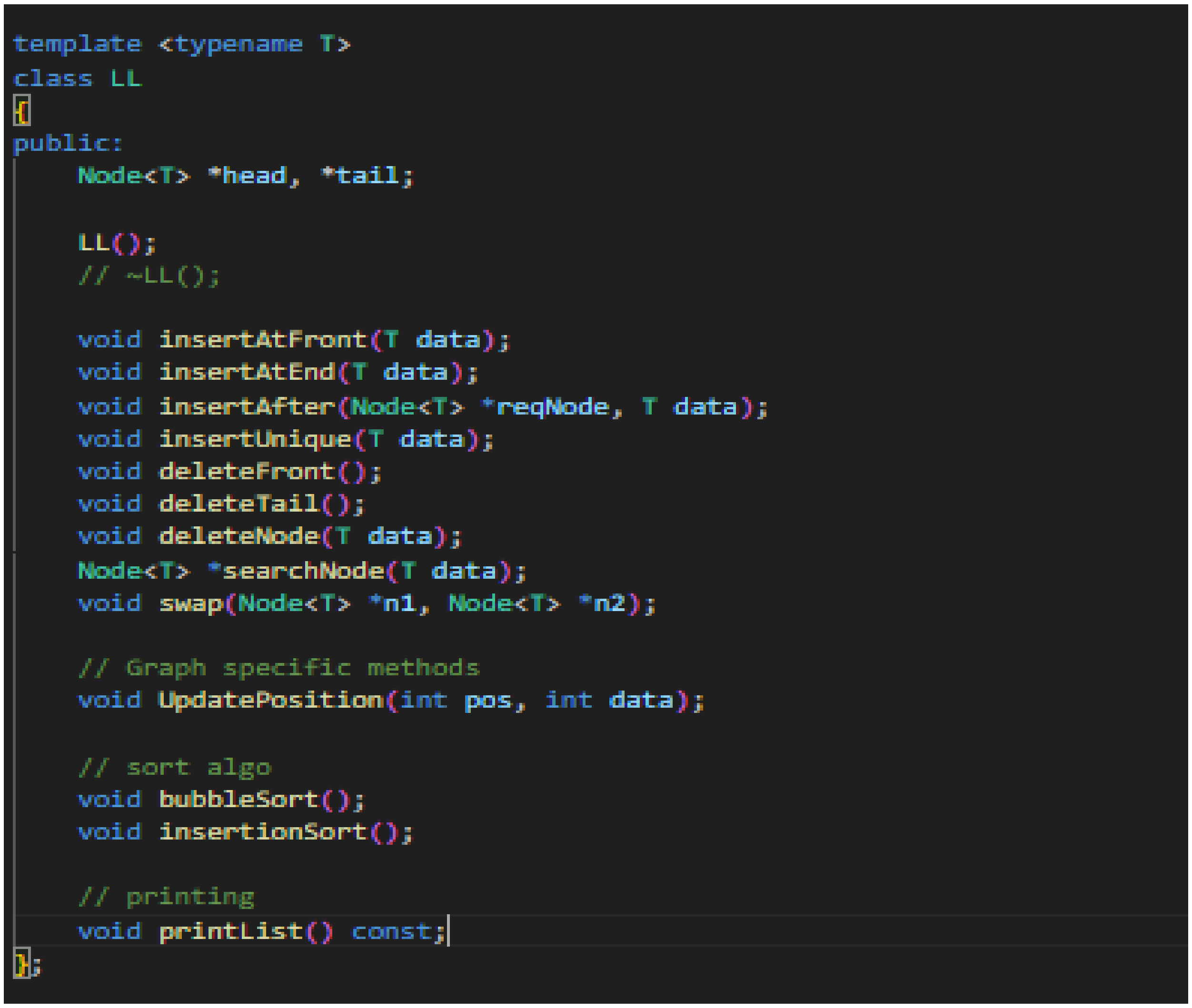
Overview

The code generates a random grid of **MxN space**, asks the user to input the starting location and provides him option of auto-completing or manually playing the Console Based 2D-Car game which we named as **Shurli.** Although playing manually gains better scores but to each their own!

Let's start with the pre-requisites. We were limited not to use the STL library and related structures e.g., vectors, lists. Now, we needed to firstly implement the linked lists (templated), queues (queues), graphs (templated), adjacency lists (templated) and lastly, max heaps for a specific structure.

Linked List (LL<T>)

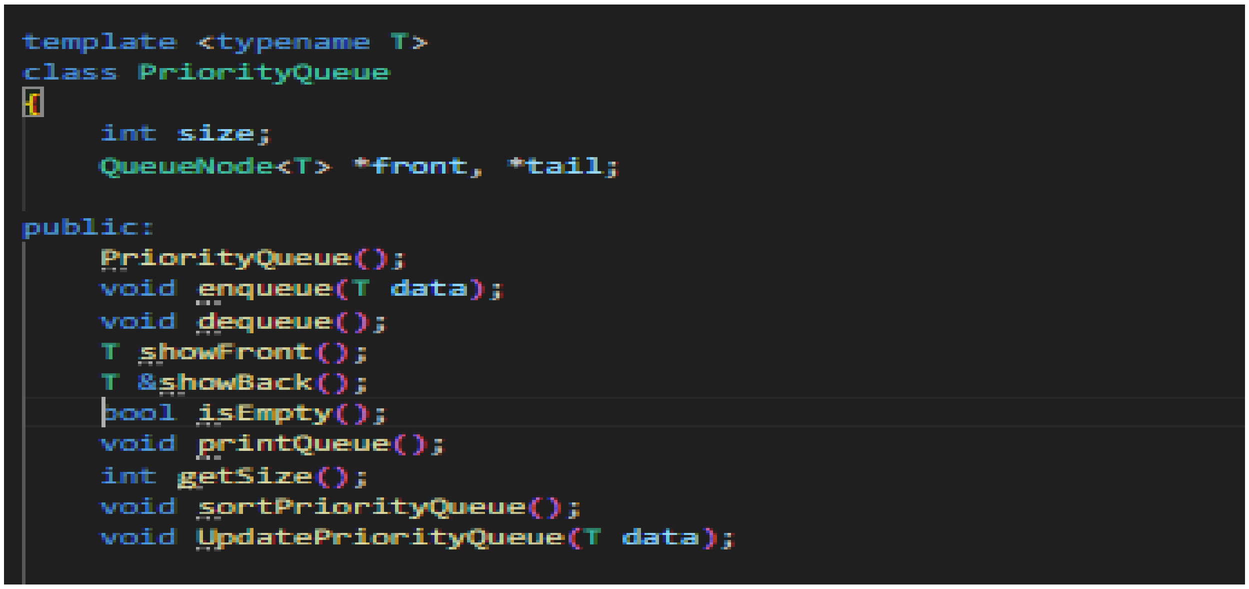
A doubly linked-list, having a tail and head pointers with the agility to traverse both forward and backward. Its implementation using templates allowed us flexibility in implementing multiple structs and classes in our code.



```
template <typename T>
class LL
{
public:
    Node<T> *head, *tail;

    LL();
    // ~LL();

    void insertAtFront(T data);
    void insertAtEnd(T data);
    void insertAfter(Node<T> *reqNode, T data);
    void insertUnique(T data);
    void deleteFront();
    void deleteTail();
    void deleteNode(T data);
    Node<T> *searchNode(T data);
    void swap(Node<T> *n1, Node<T> *n2);

    // Graph specific methods
    void UpdatePosition(int pos, int data);

    // sort algo
    void bubbleSort();
    void insertionSort();

    // printing
    void printList() const;
};
```

Queue (PriorityQueue<T>)

The queue (priority) helped us in implementing Djikstra Algorithm for finding the shortest path in a given random graph. It also being templated provided us with the flexibility of implementing the Coins, Obstacles and Power-Ups.

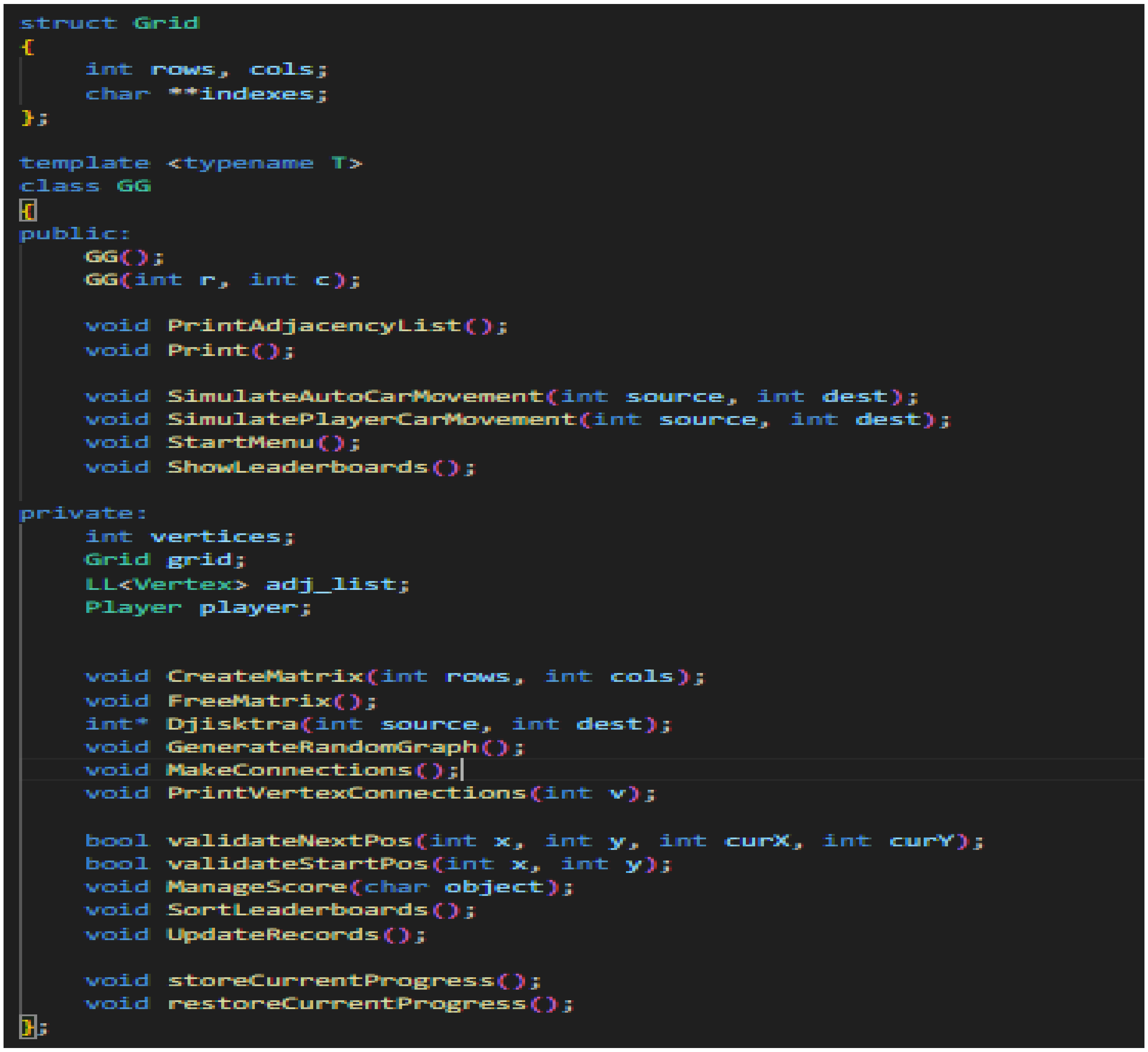

```
template <typename T>
class PriorityQueue
{
    int size;
    QueueNode<T> *front, *tail;

public:
    PriorityQueue();
    void enqueue(T data);
    void dequeue();
    T showFront();
    T &showBack();
    bool isEmpty();
    void printQueue();
    int getSize();
    void sortPriorityQueue();
    void UpdatePriorityQueue(T data);
```

Graph

Graph was probably the most fun structure to implement, its capability of doing amazing tasks such as connections, vertices and edges were fun to deal with. Also, its work integrated with adjacency lists were quite amazing to pull. The complexity of applying Djikstra to find the shorted path between the vertices and utilizing edges was really great.

Edge: This class is associated with composition in the **Vertex** class, as it contains the list of adjacent vectors in its environment.

Vertex: This class has a linked list of **Edge** class to represent its adjacent connections with the vertices. It has an ID, weight and



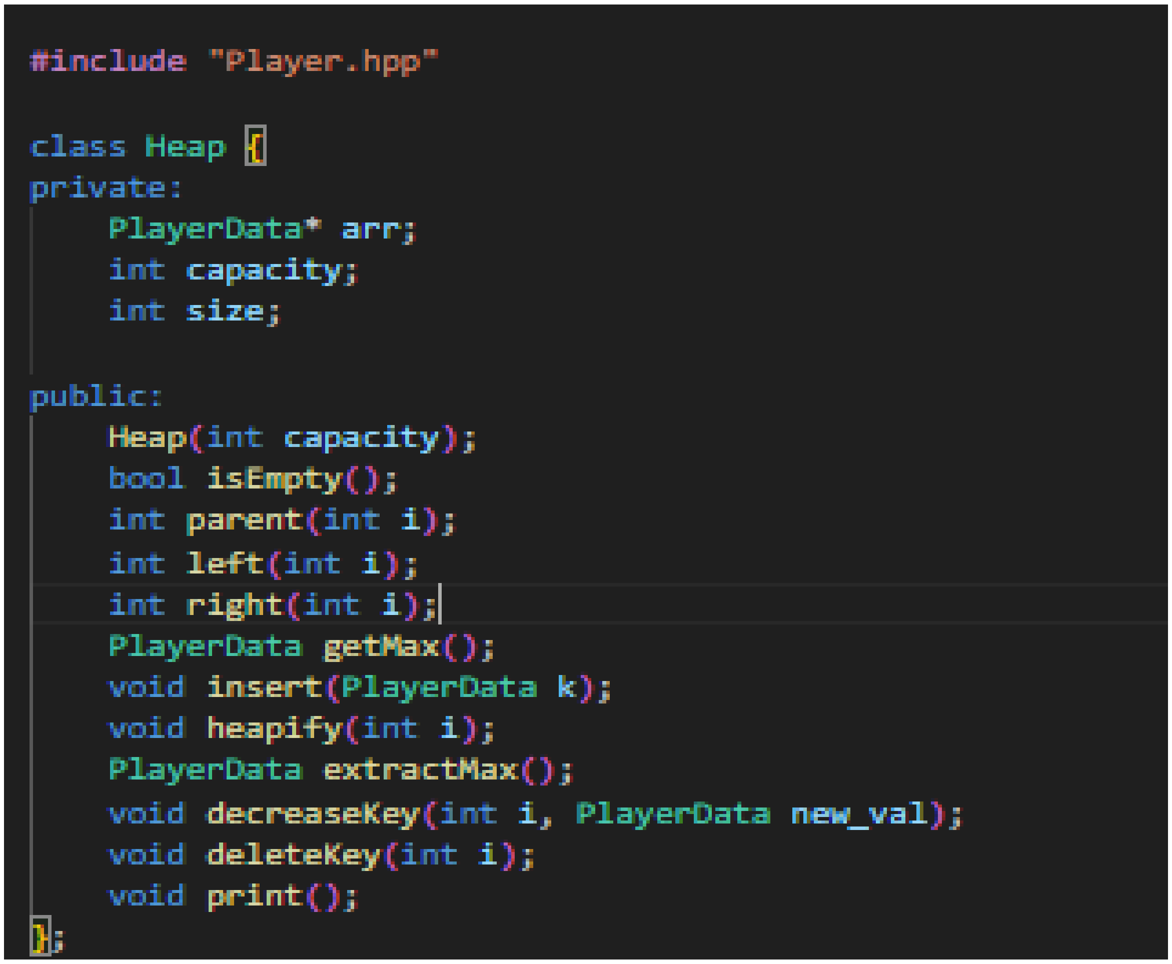
```
struct Grid
{
    int rows, cols;
    char **indexes;
};

template <typename T>
class GG
{
public:
    GG();
    GG(int r, int c);

    void PrintAdjacencyList();
    void Print();

    void SimulateAutoCarMovement(int source, int dest);
    void SimulatePlayerCarMovement(int source, int dest);
    void StartMenu();
    void ShowLeaderboards();

private:
    int vertices;
    Grid grid;
    LL<Vertex> adj_list;
    Player player;


    void CreateMatrix(int rows, int cols);
    void FreeMatrix();
    int* Djisktra(int source, int dest);
    void GenerateRandomGraph();
    void MakeConnections();
    void PrintVertexConnections(int v);

    bool validateNextPos(int x, int y, int curX, int curY);
    bool validateStartPos(int x, int y);
    void ManageScore(char object);
    void SortLeaderboards();
    void UpdateRecords();

    void storeCurrentProgress();
    void restoreCurrentProgress();
};
```

Adjacency List (LL<Vertex>)

The adjacency list has a linked list of **Vertex** objects in which each individual vertex has a linked list of **Edge** objects to show its connections.

Max Heap (Heap)

The Max Heap is implemented to sort the leaderboards and show the ranking of the players, as the Rank 1 represents the greatest among the top 10 players and Rank 10 represents the least-greatest scorer among the players.



```cpp
#include "Player.hpp"

class Heap {
private:
    PlayerData* arr;
    int capacity;
    int size;

public:
    Heap(int capacity);
    bool isEmpty();
    int parent(int i);
    int left(int i);
    int right(int i);
    PlayerData getMax();
    void insert(PlayerData k);
    void heapify(int i);
    PlayerData extractMax();
    void decreaseKey(int i, PlayerData new_val);
    void deleteKey(int i);
    void print();
};
```