COP5536 Spring 2025 Programming Project

Name : Trinesh Reddy Bayapureddy Sannala
UFID : 6264-8646
UF EMAIL : bayapureddysa.tr@ufl.edu

Project Overview

This C++ project implements a **License Plate Management System** using a **Red-Black Tree** data structure. It supports the following operations:

- Registering (custom or randomly generated) license plates.
- Deleting license plates.
- Looking up plates, as well as finding previous/next plates.
- Listing plates in a lexicographical range.
- Tracking revenue based on registrations (custom: 7 Galleons, random: 4 Galleons).

Program Structure

1. Global Definitions

- `enum Color { RED, BLACK };` — Used for Red-Black Tree node coloring.

---

2. Node Structure

```
struct Node {
    string plateNum;
    bool is_custom;
    Color color;
    Node *left, *right, *parent;
};
```

- Each node stores a license plate string, a flag if it's custom, color, and child/parent pointers.

---

3. RedBlackTree Class

Member Variables

- `Node* root` — Root of the Red-Black Tree.
- `Node* TNULL` — Sentinel node used to represent null leaves.
- `int revenue` — Tracks revenue collected from registrations.

Key Private Helper Functions

- `initializeNULLNode()` — Initializes TNULL node with black color.
- `searchTreeHelper()` — Binary search for a node.
- `inorderRange()` — Inorder traversal to collect plates between two values.
- `leftRotate()` & `rightRotate()` — Tree balancing rotations.
- `insertFix()` — Fixes Red-Black Tree property violations after insertions.
- `transplant()` — Replaces subtree for delete operations.
- `minimum()` — Finds node with minimum key.
- `deleteFix()` — Fixes tree after deletion.
- `deleteNodeHelper()` — Performs deletion.

Public Methods (Interface)

1. `RedBlackTree()`

Constructor initializes the tree and revenue system. It seeds the random generator.

2. `string addLicence(string plateNum = "")`

- Adds a license plate.
- If no plate is given, it generates a random 4-character alphanumeric plate.
- Returns a success/failure message.

3. `string dropLicence(const string& plateNum)`

- Deletes a plate from the system.
- Returns a message indicating success or failure.

4. `string lookupLicence(const string& plateNum)`

- Checks if the plate exists.
- Returns existence status.

5. `string lookupPrev(const string& plateNum)`

- Finds the lexicographically previous plate.

6. `string lookupNext(const string& plateNum)`

- Finds the lexicographically next plate.

7. `string lookupRange(const string& lo, const string& hi)`

- Returns all plates within the lexicographic range `[lo, hi]`.

8. `string revenueReport()`

- Returns a string reporting total revenue collected.

---

4. Command Processing

```
void processCommands(const string& filename)
```

- Reads commands from a file.
- Parses functions like:
    - `addLicence()`
    - `dropLicence()`
    - `lookupLicence()`
    - `lookupPrev()`
    - `lookupNext()`
    - `lookupRange(lo, hi)`
    - `revenue()`
- Writes output to a new file with `"_output.txt"` appended.

---

5. Main Function

```
int main(int argc, char* argv[])
```

- Accepts input filename from command line.
- If no argument is passed, prints usage help.
- Calls `processCommands()`.

Function Prototypes Summary

```
RedBlackTree(); // Constructor
```

```
string addLicence(string plateNum = "");
string dropLicence(const string& plateNum);
string lookupLicence(const string& plateNum);
string lookupPrev(const string& plateNum);
string lookupNext(const string& plateNum);
string lookupRange(const string& lo, const string& hi);
string revenueReport();
void processCommands(const string& filename);
int main(int argc, char* argv[]);
```

Summary

This project is an efficient and structured implementation of a **Red-Black Tree-based license plate registry system**. It simulates a real-world scenario with:

- Unique key storage,
- Fee-based services,
- Lexicographical lookups,
- Revenue tracking.