Assignment: Hash Table with AVL Tree Chaining

Objective

The goal of this assignment is to help you understand how to implement a hash table with separate chaining using **AVL trees** instead of linked lists. You will then use the hash table in a real-world application involving a **library book catalog system**.

Part 1: Data Structure Implementation

Description

Implement a hash table in C++ where:

- Each bucket uses an **AVL tree** for collision resolution.
- Keys are `std::string` (e.g., book titles or ISBNs).
- The AVL tree supports efficient operations in $\mathcal{O}(\log n)$ time per bucket.

Requirements

1. **AVL Tree**:
   - Self-balancing binary search tree.
   - Implement insertion, search and deletion.
   - Each node stores a key-value pair (`std::string`, `std::string`).
2. **Hash Table**:
   - Fixed-size array of AVL tree pointers.
   - String hash function (e.g., polynomial rolling hash or `std::hash`).
   - Public operations:
     - `void insert(const std::string& key, const std::string& value);`
     - `std::string* search(const std::string& key);`
     - `void delete(const std::string& key);`
     - `void display();` — show table contents by bucket.

Part 2: Application — Library Book Catalog

Scenario

You are building a system to store and retrieve book information by title using your hash table.

Requirements

- Each book has:
  - Title (string) — used as the key.
  - Author (string)
  - Genre (string)
- Store book data in the hash table using the title as the key.
- Value format: a combined string of author and genre (e.g., `F. Scott Fitzgerald - Fiction`).
- Load books from a file named `books.txt` formatted as:

```
The Great Gatsby,F. Scott Fitzgerald,Fiction
A Brief History of Time,Stephen Hawking,Science
...
```

- Implement a simple console menu:

```
1. Load books from file
2. Search for a book by title
3. Display hash table
4. Exit
```

Deliverables

- `AVLTree.h / AVLTree.cpp`
- `HashTable.h / HashTable.cpp`
- `main.cpp`
- `books.txt` (sample file with at least 10 entries)
- **Report (1 page)** including:
  - Description of the AVL tree.
  - Explanation of integration with the hash table.
  - Time and space complexity analysis.

Hints

- Use AVL tree balancing (LL, RR, LR, RL rotations) to maintain height.
- Balance factor = height(left) - height(right).
- Use `std::hash<std::string>` for simple hashing.

Grading Criteria

Component	Points
AVL Tree Implementation	25
Hash Table Implementation	20
File I/O and Integration	15
Search Functionality	10
Display Function	10
Code Quality & Documentation	10
Report	10