PROJECT DESCRIPTION

[COL 106 LONG ASSIGNMENT]

This project is a C++ implementation of a simplified version control system. It provides functionality for creating and managing files, taking snapshots, updating content, rolling back to earlier versions, and tracking file history. The design integrates multiple data structures—trees for organizing versions, heaps for efficiently retrieving files having largest version count or frequently updated files, and hash maps for fast indexing and lookup.

The system is organized into modular components:

* **filefunctions.cpp** handles file operations such as create, read, update, snapshot, rollback, and history.
* **filestructure.cpp** defines the core classes (File, TreeNode) and manages file metadata, content, and version details.
* **heap.cpp** manages files based on time and version counts using heap structures, supporting efficient retrieval of recent or largest versioned files.
* **map.cpp** implements hash maps for mapping file names and versions to file objects, ensuring fast access and storage. It consists of 2 maps- one version\_map for mapping version\_id to Version (This is implemented via Vector). And another map for mapping file\_name and File (This is implemented via Hashmap).
* **mainfunction.cpp, mainfunction2.cpp, and mainfunction4.cpp** serve as drivers for testing and demonstrating different functionalities. The mainfu

Overall, the project demonstrates how fundamental data structures can be applied to build a functional, modular version control system that mirrors the core ideas behind GitHub.

**filefunctions.cpp**  
This file provides the primary operations for file handling and version control. It includes:

* createFile() – Creates a new file entry in the system.
* read() – Reads the contents of a file.
* insert() – Inserts new content into a file.
* update() – Updates existing file content.
* snapshot() – Captures the current state of a file (like a commit).
* rollback() – Reverts a file back to a previous version.
* history() – Displays the version history of a file.

Classes: None defined in this file.

**filestructure.cpp**  
This file defines the structures and classes used to represent files and their versions.

* **Classes**:
  + File – Represents a file, storing its name, content, creation time, snapshots, and metadata.
  + TreeNode – Represents a node in a version tree, linking snapshots and changes.
* **Functions**:
  + commitsnapshot() – Saves a new snapshot of a file.
  + status() – Returns the current status of a file.
  + get\_version\_id() – Retrieves the version identifier of a file.
  + get\_content() – Gets the current file content.
  + change\_content() – Modifies the file’s content.
  + get\_snapshot\_time() – Returns the time a snapshot was created.
  + get\_created\_time() – Returns the file creation time.
  + getname() – Retrieves the file’s name.
  + read() – Loads file structure or metadata.

**heap.cpp**  
This file manages files using heap data structures for efficient retrieval.

* **Classes**:
  + File – Represents file metadata within heaps.
  + heapManager\_time – Manages a heap that sorts files based on last edited time.
  + heapManager\_versions – Manages a heap that sorts files based on version count.
* **Functions**:
  + addFile() – Inserts a file into the heap.
  + recentFiles() – Retrieves the most recently edited files.
  + heapify\_time\_down() / heapify\_time\_up() – Maintains heap order for time-based management.
  + updateFile\_time() – Updates a file’s timestamp in the heap.
  + removeFile() – Removes a file from the heap.
  + biggestTrees() – Retrieves the files with the largest version trees.
  + heapify\_version\_down() / heapify\_version\_up() – Maintains heap order for version-based management.
  + updateFile\_version() – Updates a file’s version count in the heap.

**map.cpp**  
This file implements hash map structures for mapping files and versions.

* **Classes**:
  + fileMap – Maps file names to their corresponding file objects.
  + versionMap – Maps version numbers to file objects.
  + TreeNode and File – Defined here as supporting structures.
* **Functions**:
  + insert() – Inserts files or versions into the map.
  + get\_size() – Returns the size of the map.
  + hashFunction() – Generates hash values for efficient indexing.
  + rehash() – Resizes and reorganizes the map when load factor exceeds limits.