Student Grade Monitoring System

PS5

Program Test Plan Document

I. Project Overview

The Student Grade Monitoring System is a command-line application designed to manage and monitor student grades efficiently. It enables adding, updating, searching, deleting, and displaying student grade records in a fast and structured way.

This system is implemented in C++, and it uses AVL Tree as its core data structure. The AVL Tree ensures that all operations: insert, delete, update and search, are performed in O(log n) time by maintaining a balanced height after every modification while the display function has the time complexity of O(n).

Problem / Needs Addressed:
Manually managing grades for multiple students can be inefficient and error prone. This project simplifies the process by maintaining student records in an automatically balanced tree that supports fast lookups, updates, and displays in sorted order by student ID.

Technical Approach:

- **Primary Data Structure:** AVL Tree (self-balancing binary search tree)
- **Interface:** Command-line interface (menu-based)
- **Persistent Storage:** CSV file storing (student_ID, name, course, grade, GPA)
- **Implementation Language:** C++

Reason for Choosing AVL Tree:
A Binary Search Tree (BST) can degrade to O(n) performance when unbalanced. The AVL Tree guarantees a balanced structure through rotations after insertions and deletions, ensuring consistent logarithmic-time performance. This makes it ideal for a project where fast and reliable grade record access is essential.

II. Unit Test Plan

In this section, the unit test plan for each method/function in the application is described. Note that ALL functions, including those in main.cpp and all class methods, are included. The argument names listed below match the actual parameter names defined in the source code.

Test ID	Unit Name	Class Name	Argument(s)	Expected Result
UT001	AVLNode (Constructor)	AVLNode	student_ID: 1, name: "John", course: "CS", grade: 90, GPA: 4.0	A new node is created with provided values. Height is 1. Left/Right pointers are null.
UT002	getStudentID	AVLNode	None	Returns the integer student_ID (e.g., 1).
UT003	getName	AVLNode	None	Returns the string name (e.g., "John").
UT004	getCourse	AVLNode	None	Returns the string course (e.g., "CS").
UT005	getGrade	AVLNode	None	Returns the float grade (e.g., 90.0).
UT006	getGPA	AVLNode	None	Returns the float GPA (e.g., 4.0).
UT007	getHeight	AVLNode	None	Returns the current height integer of the node.
UT008	setGrade	AVLNode	g: 95.5	The node's grade variable is updated to 95.5.
UT009	setGPA	AVLNode	gp: 3.8	The node's GPA variable is updated to 3.8.
UT010	getLeft	AVLNode	None	Returns the pointer to the left child node (or null).
UT011	setLeft	AVLNode	l: NodePtr	The node's left pointer is updated to point to NodePtr.
UT012	getRight	AVLNode	None	Returns the pointer to the right child node (or null).

UT013	setRight	AVLNode	r: NodePtr	The node's right pointer is updated to point to NodePtr.
UT014	AVLTree (Constructor)	AVLTree	None	Creates an empty tree with root initialized to null.
UT015	insert (Base Case)	AVLTree	student_ID: 10, name: "A", course: "CS", grade: 80	Node 10 is inserted as root. Tree height becomes 1.
UT016	insert (Recursion)	AVLTree	student_ID: 20, name: "B", course: "CS", grade: 85 (after 10)	Node 20 is inserted as right child of 10. BST property is maintained.
UT017	insert (Rotation)	AVLTree	student_ID: 30, name: "C", course: "CS", grade: 90 (after 10, 20)	Inserts 30, detects imbalance at 10, performs left rotation, and 20 becomes the new root.
UT018	search (Found)	AVLTree	student_ID: 20, course: "CS"	Returns pointer to node 20.
UT019	search (Not Found)	AVLTree	student_ID: 99, course: "CS"	Returns null indicating ID does not exist.
UT020	update (Success)	AVLTree	student_ID: 10, course: "CS", grade: 88.0	Locates node 10, updates grade and GPA, and returns true.
UT021	update (Fail)	AVLTree	student_ID: 99, course: "CS", grade: 88.0	Fails to locate node 99 and returns false.
UT022	deleteNode (Leaf)	AVLTree	student_ID: 30, course: "CS"	Removes node 30 (leaf) and returns updated root.
UT023	deleteNode (Two Children)	AVLTree	student_ID: 20, course: "CS" (Root)	Finds successor (minimum in right subtree), copies data, deletes successor, rebalances, returns new root.
UT024	display	AVLTree	None	Prints all nodes to console in ascending order of ID using inorder traversal.

UT025	saveToFile	AVLTree	filename: "data.csv"	Traverses tree and writes ID, Name, Course, Grade, and GPA to the file.
UT026	loadFromFile	AVLTree	filename: "data.csv"	Reads CSV, parses lines, and inserts nodes to rebuild a balanced tree.
UT027	getHeight (Helper)	AVLTree	node: NodePtr	Returns 0 if node is null; otherwise returns the node height.
UT028	getBalance (Helper)	AVLTree	node: NodePtr	Returns height(left) − height(right).
UT029	rotateLeft (Helper)	AVLTree	x: NodePtr	Performs left rotation around x, updates heights, and returns new subtree root.
UT030	rotateRight (Helper)	AVLTree	y: NodePtr	Performs right rotation around y, updates heights, and returns new subtree root.
UT031	FindMinNode (Helper)	AVLTree	node: NodePtr	Traverses left pointers to find the smallest ID in the subtree.
UT032	inorderTraversal	AVLTree	node: NodePtr	Recursively visits left, root, right to support the display function.
UT033	isValidName	Global	str: "John-Doe"	Returns true (contains only letters, spaces, hyphens).
UT034	isValidName	Global	str: "John123"	Returns false (contains digits).
UT035	isValidCourse	Global	str: "CSC 307"	Returns true (contains alnum, space).
UT036	isValidCourse	Global	str: "CSC@307"	Returns false (contains special char '@').
UT037	getIntInput	Global	prompt: "Enter ID", min, max	Reads valid integer from input stream; reprompts on invalid input.

UT038	getFloatInput	Global	prompt: "Enter Grade", min, max	Reads valid float from input stream; reprompts on invalid input.
UT039	getStringInput	Global	prompt: "Name", isName: true	Reads non-empty string matching name criteria.
UT040	deleteAllCourses	AVLTree	student_ID: 10	Removes all nodes associated with student ID 10.
UT041	getStudentName	AVLTree	student_ID: 10	Returns the name string associated with student ID 10.
UT042	displayStudent	AVLTree	student_ID: 10	Searches for and prints all course records for student ID 10.
UT043	getAllCourses	AVLTree	student_ID: 10	Returns a std::vector containing pointers to all nodes for student ID 10.
UT044	setStudentID	AVLNode	id: 5	Updates the node's student_ID to 5.
UT045	setName	AVLNode	n: "Alice"	Updates the node's name to "Alice".
UT046	setCourse	AVLNode	c: "Math"	Updates the node's course to "Math".
UT047	setHeight	AVLNode	h: 2	Updates the node's height to 2.

III. Integration Test/Acceptance Test

This section defines the integration tests ensuring interconnected components and system commands work as required.

Test ID	Test Title	Test Description	Expected Result
IT001	Add Student Command	This test determines if the "Add" command correctly takes user input and stores it in the AVL Tree.	Command successfully adds a student record. The "Display" command confirms the student appears in the list.
IT002	Search Student Command	This test determines if the "Search" command can locate a record added in IT001 using the Student ID.	Command output displays the correct Name, Course, Grade, and GPA for the requested ID.
IT003	Update Grade Command	This test determines if the "Update" command correctly modifies the grade and GPA of an existing student.	Command updates the record. A subsequent "Search" shows the new Grade and GPA, not the old ones.
IT004	Delete Student Command	This test determines if the "Delete" command removes a student and maintains the sorted structure.	Command removes the student. A subsequent "Search" for that ID results in "Not Found."
IT005	Display All Command	This test determines if the "Display" command prints all records sorted by Student ID.	Command outputs the full list of students strictly ordered by ID (Ascending).
IT006	Persistence (Save)	This test determines if the "Save" command correctly writes the current tree state to a CSV file.	A CSV file is created on the disk containing accurate data for every student currently in the tree.
IT007	Persistence (Load)	This test determines if the "Load" command correctly reconstructs the AVL tree from a CSV file.	Upon loading, the program memory is populated with records from the file, and the tree is automatically balanced.
IT008	Data Integrity	This test combines Load and Display to ensure no data is lost during File I/O.	The total number of records displayed after "Load" matches the number of records present before "Save."

IT009	Invalid Input Handling	This test checks if the system handles searching for a non-existent ID gracefully.	The system displays a user-friendly error message and does not crash or exit unexpectedly.

IV. Test Results

The following table logs the execution of every test defined in Section II and III.

Test ID	Date/Time Completed	Actual Result	Pass/Fail?
UT001	Nov 24, 2025, 10:00 AM	Node created with ID 1, Name "John", and correct fields.	Pass
UT002	Nov 24, 2025, 10:01 AM	Returned ID 1.	Pass
UT003	Nov 24, 2025, 10:01 AM	Returned Name "John".	Pass
UT004	Nov 24, 2025, 10:01 AM	Returned Course "CS".	Pass
UT005	Nov 24, 2025, 10:01 AM	Returned Grade 90.0.	Pass
UT006	Nov 24, 2025, 10:01 AM	Returned GPA 4.0.	Pass
UT007	Nov 24, 2025, 10:01 AM	Returned Height 1.	Pass
UT008	Nov 24, 2025, 10:05 AM	Grade updated to 95.5.	Pass
UT009	Nov 24, 2025, 10:05 AM	GPA updated to 3.8.	Pass
UT010	Nov 24, 2025, 10:05 AM	Returned Null (correct for new node).	Pass
UT011	Nov 24, 2025, 10:05 AM	Left pointer updated to new address.	Pass
UT012	Nov 24, 2025, 10:05 AM	Returned Null (correct for new node).	Pass
UT013	Nov 24, 2025, 10:05 AM	Right pointer updated to new address.	Pass
UT014	Nov 24, 2025, 10:10 AM	Tree created with root = null.	Pass

UT015	Nov 24, 2025, 10:15 AM	Node 10 inserted. Root is 10.	Pass
UT016	Nov 24, 2025, 10:15 AM	Node 20 inserted to Right of 10.	Pass
UT017	Nov 24, 2025, 10:20 AM	Node 30 inserted. Root rotated to 20. Balanced.	Pass
UT018	Nov 24, 2025, 10:25 AM	Search(20) returned valid pointer.	Pass
UT019	Nov 24, 2025, 10:25 AM	Search(99) returned null.	Pass
UT020	Nov 24, 2025, 10:30 AM	Node 10 updated. Search confirms new values.	Pass
UT021	Nov 24, 2025, 10:30 AM	Update(99) returned false.	Pass
UT022	Nov 24, 2025, 10:35 AM	Node 30 deleted. Tree structure valid.	Pass
UT023	Nov 24, 2025, 10:40 AM	Root deleted. Successor replaced root correctly.	Pass
UT024	Nov 24, 2025, 10:45 AM	Display showed nodes in sorted ID order.	Pass
UT025	Nov 24, 2025, 10:50 AM	"data.csv" file created with text content.	Pass
UT026	Nov 24, 2025, 10:55 AM	Tree rebuilt from "data.csv" correctly.	Pass
UT027	Nov 24, 2025, 11:00 AM	Returned correct height integers.	Pass
UT028	Nov 24, 2025, 11:00 AM	Returned correct balance factors (0, 1, -1).	Pass
UT029	Nov 24, 2025, 11:05 AM	Left rotation logic verified correct.	Pass
UT030	Nov 24, 2025, 11:05 AM	Right rotation logic verified correct.	Pass

UT031	Nov 24, 2025, 11:10 AM	Min node identified correctly.	Pass
UT032	Nov 24, 2025, 11:15 AM	Traversal visited nodes in correct sequence.	Pass
UT033	Nov 24, 2025, 11:20 AM	Returned true for valid name input.	Pass
UT034	Nov 24, 2025, 11:20 AM	Returned false for name with digits.	Pass
UT035	Nov 24, 2025, 11:25 AM	Returned true for valid course input.	Pass
UT036	Nov 24, 2025, 11:25 AM	Returned false for course with special characters.	Pass
UT037	Nov 24, 2025, 11:30 AM	Correctly handled non-integer input and accepted valid int.	Pass
UT038	Nov 24, 2025, 11:35 AM	Correctly handled non-float input and accepted valid grade.	Pass
UT039	Nov 24, 2025, 11:40 AM	Accepted valid string and rejected empty input.	Pass
UT040	Nov 24, 2025, 11:45 AM	All courses associated with ID removed successfully.	Pass
UT041	Nov 24, 2025, 11:50 AM	Correct name string returned for ID.	Pass
UT042	Nov 24, 2025, 11:55 AM	Student specific records displayed correctly.	Pass
UT043	Nov 24, 2025, 12:00 PM	Vector returned with correct count of nodes.	Pass
UT044	Nov 24, 2025, 12:05 PM	ID field updated successfully.	Pass
UT045	Nov 24, 2025, 12:05 PM	Name field updated successfully.	Pass
UT046	Nov 24, 2025, 12:05 PM	Course field updated successfully.	Pass

UT047	Nov 24, 2025, 12:05 PM	Height field updated successfully.	Pass
IT001	Nov 24, 2025, 01:00 PM	User Input added student successfully.	Pass
IT002	Nov 24, 2025, 01:10 PM	Found student and printed details to screen.	Pass
IT003	Nov 24, 2025, 01:20 PM	Grade change persisted in memory.	Pass
IT004	Nov 24, 2025, 01:30 PM	Student removed. ID no longer found.	Pass
IT005	Nov 24, 2025, 01:40 PM	Full list printed 10, 20, 30...	Pass
IT006	Nov 24, 2025, 01:50 PM	CSV file on disk matches tree data.	Pass
IT007	Nov 24, 2025, 02:00 PM	Program restarted, data loaded back perfectly.	Pass
IT008	Nov 24, 2025, 02:10 PM	Record count before Save equals count after Load.	Pass
IT009	Nov 24, 2025, 02:15 PM	"Student Not Found" printed. App stayed open.	Pass

V. Member Contributions

The group members collaborated to ensure all components of the system were thoroughly tested.

- **Prashant Chand:** Designed the Unit Test Plan for the core AVLTree structure and the insert logic (UT014-UT017), ensuring the self-balancing properties were verified.
- **Sambhav Pyakurel:** Developed the Integration Tests for File I/O (IT006-IT008) and verified the display and search operations.
- **Sujal Maharjan:** Responsible for logging the Test Results (Section IV) and developing tests for the critical modification operations, specifically delete and update (UT020-UT023, IT003-IT004).