CS2263 Final Project Report

STUDENT GRADE MANAGEMENT SYSTEM

TEAM - 21

1. Summary of Features Implemented

This project is a console-based Student Grade Management System, built in C, consisting of features for managing student records and grades.

Core Features:

- Student Management: Add, display, remove, and search students by ID or name
- Grade Management: Add and display grades for specific students
- Student Performance: Calculate GPA, letter grades, and display a visual bar chart
- Data Persistence: Save and load student and grade data to/from CSV files
- Sorting Capability: Sort students by ID, name, or GPA

Enhanced Features:

- Data Validation: Robust validation of student IDs, names, course names, and marks
- Visual Representation: Color-coded bar chart to visualize student performance
- Memory Management: Dynamic memory allocation and deallocation for linked lists

2. Use of Key Programming Concepts

- **Linked Lists**: Two separate linked lists are used for storing student information and grade data.
- **File I/O**: CSV files store student and grade data for persistence between program runs
- **Dynamic Memory Allocation**: Memory is allocated and freed for each node in the linked lists
- **Structures**: Custom data structures for students and grades
- **Sorting Algorithms**: sorting implementation for linked lists

3. Memory Management

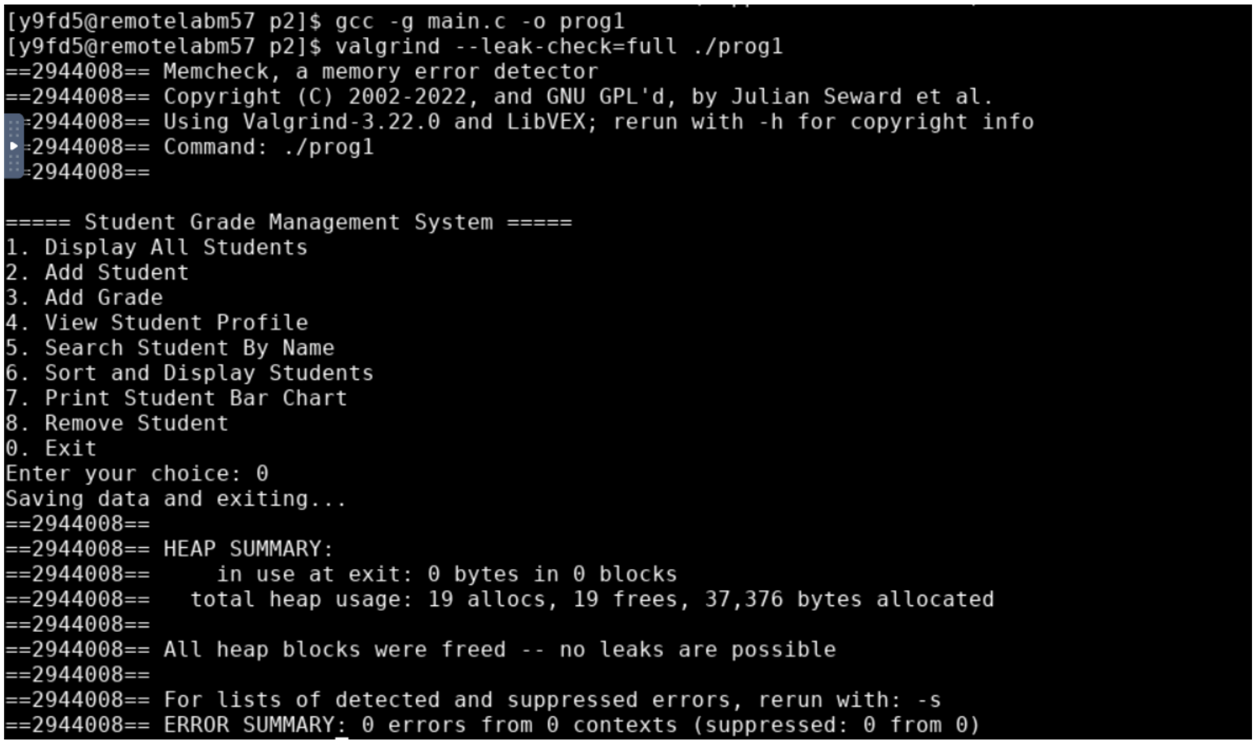
Dynamic memory is allocated when creating new student or grade records

Memory is properly freed when removing students or when the program exits.

Memory Safety:

- Null checks after memory allocation
- Proper freeing of allocated memory
- Prevention of memory leaks through recursive free functions

The program was tested with Valgrind in a Linux environment. No memory leaks were observed.



```
[y9fd5@remotelabm57 p2]$ gcc -g main.c -o prog1
[y9fd5@remotelabm57 p2]$ valgrind --leak-check=full ./prog1
==2944008== Memcheck, a memory error detector
==2944008== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==2944008== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==2944008== Command: ./prog1
==2944008==

===== Student Grade Management System =====
1. Display All Students
2. Add Student
3. Add Grade
4. View Student Profile
5. Search Student By Name
6. Sort and Display Students
7. Print Student Bar Chart
8. Remove Student
0. Exit
Enter your choice: 0
Saving data and exiting...
==2944008==
==2944008== HEAP SUMMARY:
==2944008==     in use at exit: 0 bytes in 0 blocks
==2944008==   total heap usage: 19 allocs, 19 frees, 37,376 bytes allocated
==2944008==
==2944008== All heap blocks were freed -- no leaks are possible
==2944008==
==2944008== For lists of detected and suppressed errors, rerun with: -s
==2944008== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

4. Testing and Validation

Data validation is implemented for all user inputs:

- Student ID validation (must be between 1000-10000)
- Name validation (alphabetic characters and spaces only)
- Course name validation (must be between 0-7 characters)
- Marks validation (must be between 0-100)

Error handling:

- Duplicate student ID detection
- Non-existent student ID handling
- File opening/writing error handling
- Memory allocation failure handling

The program handles key edge cases:

Empty Student Lists:

- Displays appropriate messages when lists are empty
- Prevents operations on empty lists from causing crashes

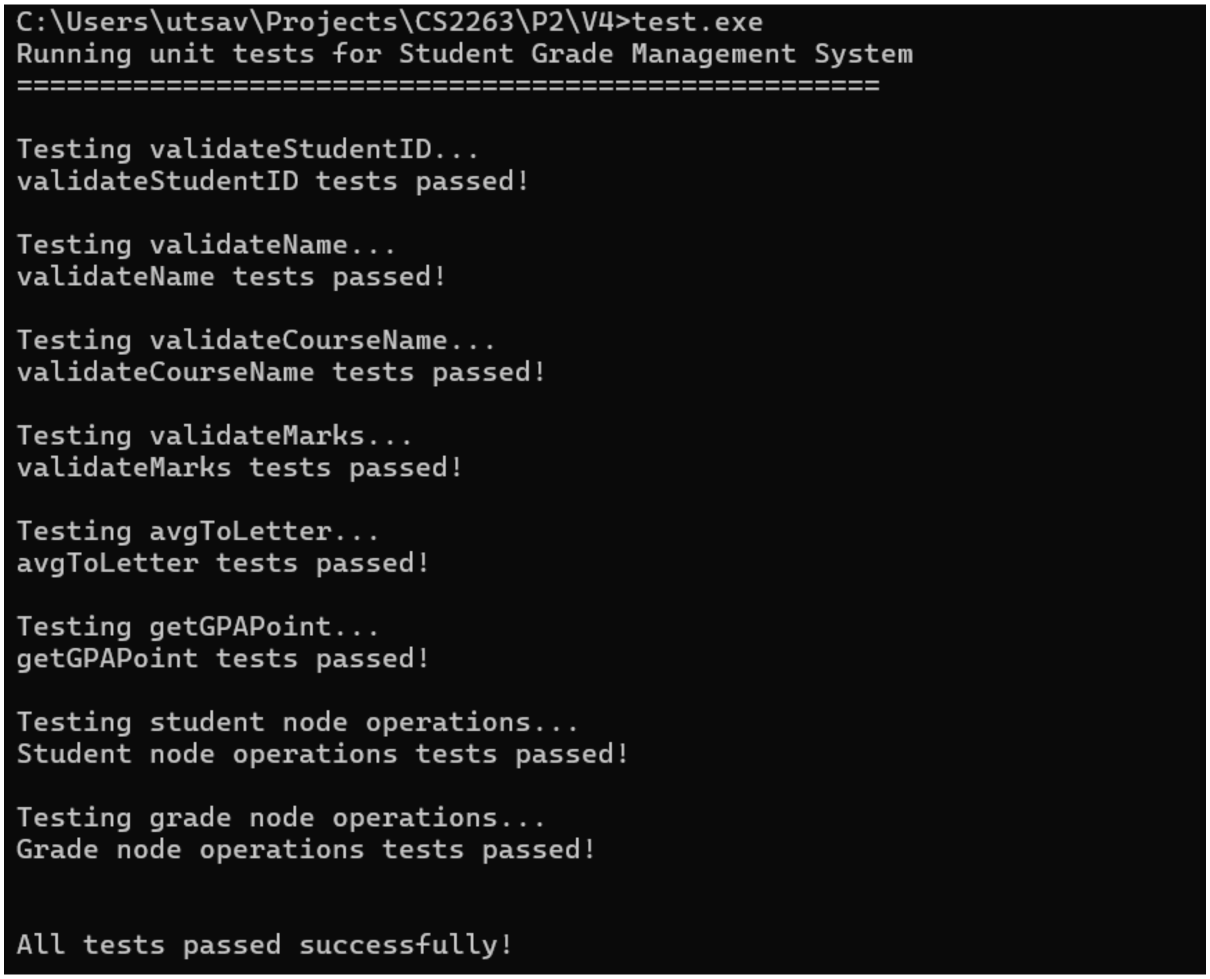
Duplicate Entries:

- Enforces student ID uniqueness via validation checks
- Returns clear error messages when duplicate IDs are detected

Extreme GPA Values:

- Handles the full range of marks (0-100)
- Correctly assigns letter grades and GPAs for all values
- Properly visualizes extreme values in the bar chart system

All these are tested by test.c; which tends to test numerous functions of the Program, including validation functions, linkedList operations, and GPA calculation.



```
C:\Users\utsav\Projects\CS2263\P2\V4>test.exe
Running unit tests for Student Grade Management System
=====================================================

Testing validateStudentID...
validateStudentID tests passed!

Testing validateName...
validateName tests passed!

Testing validateCourseName...
validateCourseName tests passed!

Testing validateMarks...
validateMarks tests passed!

Testing avgToLetter...
avgToLetter tests passed!

Testing getGPAPoint...
getGPAPoint tests passed!

Testing student node operations...
Student node operations tests passed!

Testing grade node operations...
Grade node operations tests passed!


All tests passed successfully!
```

5. Challenges Faced and Lessons Learned

Technical Challenges:

- We lost a lot of time trying to fix a linking error that kept showing up (even in a previous assignment). We tried at least 10 things, but nothing worked. That's why all our code is in a single main.c file instead of being split up. Sorry if it's a bit hard to read!
- We started this project early during reading break, but later had to switch everything to use linked lists since that was taught near the end. This change completely altered our program's logic, and redoing it under time pressure made us miss the deadline.
- Using color in the bar chart wasn't really hard—it was actually fun to learn how C can print in color using ANSI codes. It turned out great!

Lessons Learned:

- Linked lists provide flexible dynamic storage
- Input validation is critical for overall data integrity
- File I/O provides simple but effective and easy to use
- ANSI color codes can significantly improve user experience in console applications