CSC 365 - Lab 1a Write-Up

Team Members

- Stephen Chen
- Belal Elshenety

Initial Decisions

- **Programming Language:** Python
- **Libraries Used:** None (Python standard library only)
- **Environment:** Developed and tested through our MacBook terminal, ensuring no command line arguments are needed.

Architecture

Core Program Design

1. **File Parsing:**
    - The program reads `students.txt` and parses it into a list of dictionaries, where each dictionary represents a student.
    - Error handling ensures graceful termination if the file is missing or malformed.
2. **Search Functions:**
    - Separate functions handle each search operation:
        - `search_students_by_last_name`: Searches for students by last name, with optional bus information.
        - `search_students_by_teacher`: Finds students taught by a specific teacher.
        - `search_students_by_bus`: Retrieves students on a specific bus route.
        - `search_students_by_grade`: Lists all students in a specified grade.
        - `grade_high_low_gpa`: Finds the student with the highest or lowest GPA in a grade.
        - `grade_average_gpa`: Calculates the average GPA for a grade.

        - `info_summary`: Summarizes the number of students per grade.
3. **Main Program Flow:**
    - Continuously prompts the user for commands.
    - Parses the input to identify valid commands and executes the appropriate function.
    - Handles both short (e.g., `S:`) and long forms (e.g., `Student:`) of commands.
    - Gracefully skips invalid commands, comments, and empty lines.
4. **Error Handling:**
    - Ensures invalid inputs (e.g., non-numeric grades) are handled gracefully.
    - Skips comments (`//`) and blank lines in the input.
5. **Testing:**
    - The `grep` command is used to filter comments and blank lines from the test cases when running the program, ensuring only valid commands are processed.

Task Log

Stephen Chen

- Wrote all the core functions for the program, including:
    - `load_students`: Parses `students.txt` and loads the data into a structured format.
    - `search_students_by_last_name`: Retrieves students by last name, with an option to include bus route details.
    - `search_students_by_teacher`: Finds students taught by a specified teacher.
    - `search_students_by_bus`: Lists students on a specific bus route.
    - `search_students_by_grade`: Retrieves all students in a given grade.
    - `grade_high_low_gpa`: Determines the student with the highest or lowest GPA in a specific grade.
    - `grade_average_gpa`: Calculates the average GPA for students in a grade.
    - `info_summary`: Summarizes the number of students in each grade.
- Implemented the main program flow to handle user input and execute commands based on short and long forms.
- Ensured robust error handling and seamless execution for all program requirements.

Belal Elshenety

- Modified the main program to handle both short and long forms of commands, ensuring consistent user input handling.

- Wrote the `tests.txt` file with 52 test cases to cover all program requirements comprehensively.
- Modified `README.md` to include detailed instructions for running tests and converted it to a Markdown file for clarity.
- Wrote the `grep` command to filter out comments and blank lines from `tests.txt` during testing, ensuring only valid commands are processed.

Testing Notes

- **Issues Found:**
  - Initial implementation only supported short forms of commands, resulting in errors for long-form inputs.
  - The program attempted to execute comments and blank lines in `tests.txt`, causing errors during testing.
- **Resolutions:**
  - Long-form command support was added to the main program.
  - Comments and blank lines were filtered out using `grep` when running tests.
- **Time Spent on Testing:**
  - Approximately 2 hours.
- **Test Cases:**
  - A total of 52 test cases were written and executed, covering all requirements and edge cases.
  - The `tests.txt` file includes expected outputs for all commands.

Final Notes

- **Challenges:**
  - Ensuring consistent formatting across both short and long forms of commands.
  - Handling edge cases, such as missing or malformed input files and invalid commands.
- **Future Improvements:**
  - Add a `Help` command to display valid commands and usage examples.
  - Automate the test case filtering process (e.g., ignoring comments and empty lines) within the program itself, eliminating reliance on external tools like `grep`.
- **Summary:**
  - The program meets all specified requirements (R1–R13) and has been rigorously tested to ensure compliance.

- The test suite (`tests.txt`) and corresponding output (`tests.out`) demonstrate the program's correctness and robustness.