Spreadsheet Engine - C Lab

Prakhar Gupta 2023CS10493
Keshav Bansal 2023MT110273
Dhruv Pawar 2023CS5230184

March 2, 2025

1 Introduction

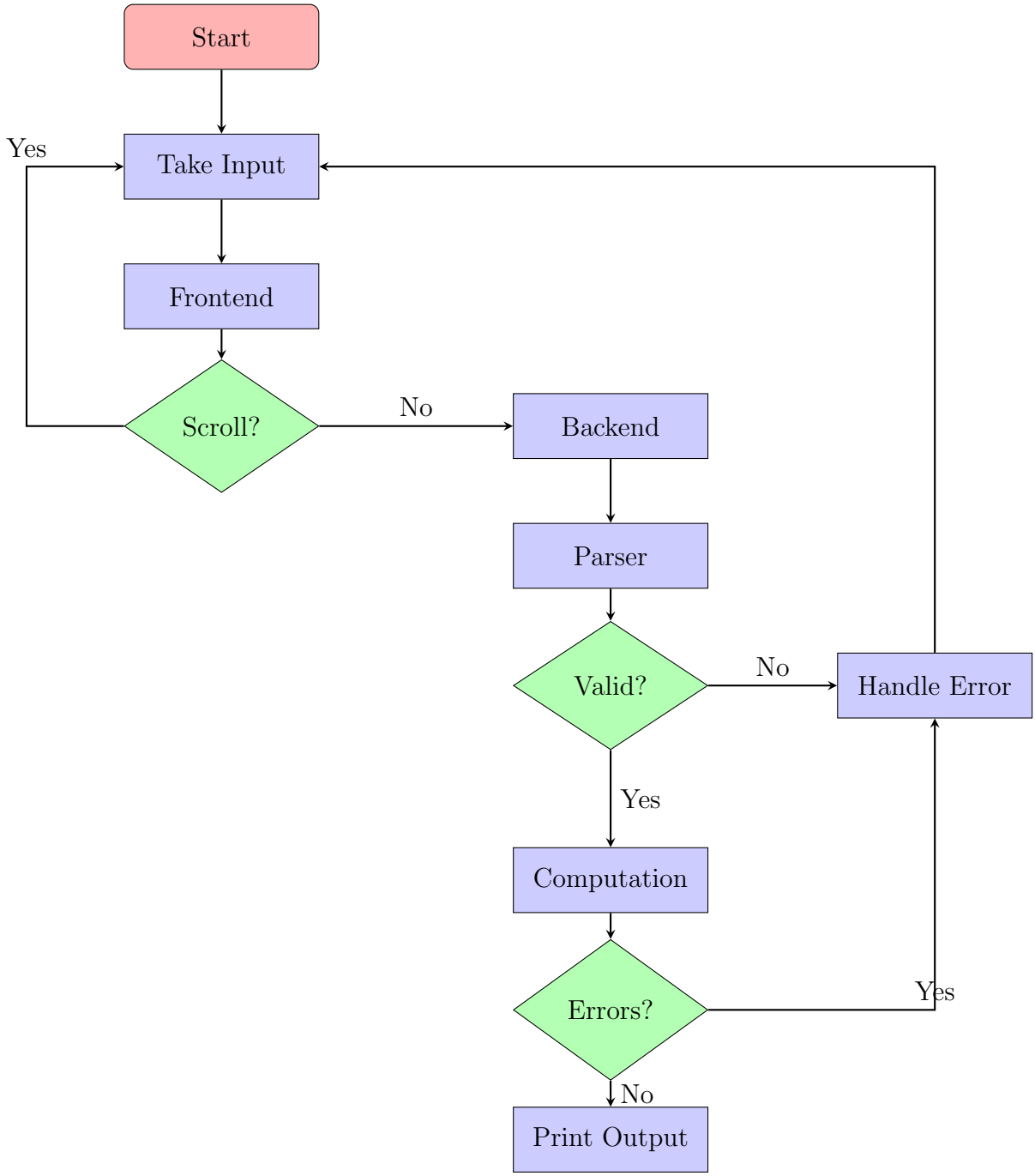
This report provides an overview of the Spreadsheet Engine we have designed for the COP290 C lab. The program supports user interactions with a simple command-line interface and performs major Excel functionalities to a great extent.

2 Structure

The project follows a modular structure with key files integrating to build the whole project. Each file has its own function following a workflow, ensuring proper data handling and maximum efficiency.

2.1 Key Files

- **main** - Entry point of the application.
- **parser** - Translates user inputs with valid checks.
- **frontend** - Maintains the interface.
- **backend** - Handles computation and data storage.
- **structs, cell, and vec** - Various utility functions and data structures.
- **test directory** - Various tests handling all edge cases and functionalities, ensuring correctness.
- **README** - Short summary and guide for using the project.
- **Makefile** - Automates compilation and execution in the terminal.



2.2 Features and Functionalities

- **Scrolling** - Navigate using 'w' (up), 'd' (right), 'a' (left), and 's' (down) by 10 cells until the spreadsheet edges.
- **Enable/Disable** - Toggle to display the virtual port.
- **Sleep** - Pauses execution for x seconds (for testing).
- **Operations** - Handles basic and range based (no floating-point arithmetic). Instructions like A1 = B1 are translated into A1 = B1 + 0

2.3 Graph and Algorithms

- **Dependency Graph** - A directed graph where each cell is a node, ensuring updates propagate efficiently while detecting circular dependencies.
- **Topological Sorting** - Ensures updates are processed in dependency order using a stack-based approach, preventing redundant calculations. Each cell tracks its dependencies and updates only after all parent cells are updated by maintaining dirty parents.

3 Test Cases and Edge Scenarios

The test suite verifies correctness across various edge cases.

3.1 Edge Cases Considered

- Invalid inputs like "hello", "A1 + B2 = B3", or "A1 = 2+3+4" are discarded.
- Large-scale range operations optimize recalculations using topological sorting.
- Handling computational errors gracefully.

3.2 Error Handling

The frontend prints [err] instead of [ok] in case of errors.

- We have incorporated an error element in each cell during initialization that ensures proper handling and error propagation in case of numerous dependencies. The cells having an error are assigned the value 0 as a place holder.
- The parser issues proper tokens to weed out any invalid inputs.
- Errors like Circular dependency are located through graph algorithms and the query is rejected, no updation in the graph is done.

4 Conclusion

This report summarizes the design, testing, and robustness of our project. For hands-on experience, visit our repository.

5 Links

- GitHub Repository: Project Repository
- Demo Video: Watch the Demonstration