COP290 Semester 2024-2025 Sem II

Rust Lab: Spreadsheet program

In the Rust lab, you will first re-implement the spreadsheet program in rust that you already implemented in C. You need to ensure that you maintain the *exact* same interface. We will use the same autograder on your Rust lab too!

You will however do some non-trivial extensions to the spreadsheet program. Here, we make one suggestion for making such extensions.

Hacker Spreadsheet

In this extension, you can design a more usable text-based spreadsheet editor (think vim for spreadsheets). While the default behaviour is autograder compatible, you can accept command-line flags to enable these extensions (e.g., ./sheet –vim).

Motivation: Hackers prefer the terminal as they believe touching the mouse takes their hands off the keyboard; slowing them down. If a spreadsheet is on a remote machine, they can ssh to the machine and edit it directly with your editor instead of having to download it (like for .xlsx). Hackers also like privacy; they may not want to give their spreadsheet to Google or Microsoft programs.

You can have modes like vim's insert mode, command mode, and normal modes.

1. *Navigation*: Opening the spreadsheet puts a *cursor* at A1. Users can move the cursor like vim: `h` for left, `j` for below, `k` for above, `l` for right. Users can jump the cursor like vim jumps file number. Do `:ZZ99` to scroll to cell ZZ99. Cursor always stays on a cell. When a cursor is on the cell, a status bar shows the formula for the cell.
2. *Insert mode*: Just like vim, pressing `i` puts us in insert mode. We can put a value or a formula in a cell. Pressing `Esc` brings us out of insert mode.
3. *Cut (d) copy (y) paste (p)* just like vim. dr (yr) cuts (copies) the whole row. dc (yc) cuts (copies) the whole column. 5dr (5yr) cuts (copies) 5 rows, etc. Cutting a row (column) moves the spreadsheet up (left). How to manage formulas when we are moving cells around? How should the program allocate the sheet in the first place?
4. *Select mode:* Just like vim, pressing v puts the user in select mode. Users can select any 2D range and press y to copy it. How is the sheet adjusted if the user presses d on a 2D range?
5. *Search:* Just like vim, you can search forwards (/), backwards (?), go to the next match (n), go to the previous match (N), or search and replace (:%s).
6. *Formulas:* Support floats and formulas like average, percentile, stddev. You can also support specifying formulas for multiple cells. For example, "`:i in 1..10: Ai = Bi + 1`" sets formulas for A1 to A10.
7. *Opening and saving files:* Your program opens standard formats like .csv and .tsv which just have values (like `./sheet myfile.csv`) and can save it into a custom format that supports formulas (`:wq myfile.ss`). How to save formulas?

You can think about how the spreadsheet works smoothly with other terminal programs. For example, one can pipe any content in stdout to open in your program (similar to `vim -`).

We've given the above just as an example extension. We prefer that you design your own extensions. You are free to extend it however you want. Other examples are implementing a collaborative browser-based Google-like or Airtable-like spreadsheet.

Timeline

Proposal deadline: ~~16 March 2025.~~ 23 March 2025. 11:59pm.

1. Declare your three-person team here. You can choose to stick to your C Lab team. Or you can create a different team. There are no section A / section B restrictions for the Rust lab.
2. Submit a design document similar to the C Lab document on Moodle. Also specify your team in the document. Please ensure that only one member in your team submits.
3. Try to *exactly describe* what your spreadsheet will do. Typically, in industry, product managers talk to customers to write *product requirements documents* (PRDs) and share them with software engineers. You are encouraged to learn how PRDs are written.
4. You are encouraged to learn wireframing softwares like Figma to define the look and feel of your system.
5. You should try to define key interfaces in this phase. For example, if you plan to build a spreadsheet with a browser GUI, what APIs does the backend expose to the frontend? Does the backend use a database? What is the schema of the database? How do you serialize a spreadsheet into a file? You are encouraged to learn modeling languages like state charts and UMLs.
6. You should try to define how exactly you will test your system. What kind of load tests, stress tests, unit tests, integration tests, etc will you run? How often will you run them? We will expect you to set up a Continuous integration system via Github Actions.

More details on final submission will follow soon. No extensions are allowed!

Final deadline: 25 Apr 11:59pm.

Subject to change based on major exam schedule.
Deliverables: TBD