ELEC 278 Assignment (S001 and S002)

Nick Mertin, BASc, EIT

September 24, 2023

1 Introduction

This is the only assignment in the course; it is worth 12 % of your final grade. It is an opportunity for you to apply what you are learning in the course to a practical software engineering problem. You are expected to complete all parts of the assignment *individually*, though discussion of the problem itself is encouraged; the OnQ discussion forum will (hopefully) be a good venue for this.

The goal of the assignment is to implement data structures and algorithms to store and manage the contents of a spreadsheet, such as one you might edit with Microsoft Excel. You will need to implement a much smaller feature set than Excel; your application only needs to be able to store text and numbers, and to evaluate equations that consist of adding numbers and cells (the exact requirements are set out in §3). Moreover, we have provided you with starter code that takes care of displaying a table and interacting with the user; that is outside of the scope of this course anyways! All you have to do is figure out how you are going to represent the data in memory, and implement certain functions which the starter code expects to be implemented.

The assignment assesses your abilities with respect to CLOs 3, 4, 5, and 6. You are expected to not just write code, but to approach this as a complete engineering problem; you must be able to propose and justify a design, implement it correctly, provide documentation on the design and implementation, and demonstrate that the implementation works as intended. The deliverables consist of the finished code for the spreadsheet application and a brief engineering report containing the associated documentation (the exact expectations are set out in §4).

2 Academic Integrity and Generative Artificial Intelligence

The use of generative artificial intelligence (AI) tools, such as ChatGPT and Copilot, is not outright banned; however, use of these tools in place of your own technical work could constitute a departure from academic integrity. Some specific examples, ranging from completely acceptable to unacceptable, are:

- Not using generative AI tools: this is always a valid approach and is your safest bet.
- Asking a tool to identify grammar errors in a report: this is completely fine, as it does not affect the technical work you are demonstrating other than potentially making the marking TAs' jobs easier by making the writing clearer.
- Asking a tool to summarize this handout for you: this is definitely not an academic integrity problem, but do at your own risk; this document is the authority on the assignment requirements, regardless what ChatGPT decides is unimportant!
- Asking a tool to help you structure your report, given content that you wrote: this is probably okay *in the context of this assessment*, since we are specifically evaluating your ability to make and justify engineering decisions. In other courses this may not be acceptable.
- Asking a tool for ideas on where to start on a task: this is entering a grey area, since it is unclear how much of the problem-solving the tool would be willing to do for you. Also, it may not give the right answer! Remember that it is *your* job to solve the challenge presented in this assignment!

- Asking a tool to solve parts of the problem for you or generate parts of the report for you: this is considered a departure from academic integrity.

Note: generative AI tools have a lot of inherent information which they leverage to perform their tasks. Therefore, **if you use any generative AI tools for any part of the assignment, you must cite them appropriately** as you would any source.

We will be keeping an eye on what is submitted and doing our best to enforce these guidelines. A key question you can ask yourself when trying to figure out whether it is acceptable to use a generative AI tool for a particular part of the assignment is: if I asked my TA this question, would they be willing to answer it? If not, then it's probably not acceptable for the AI tool to answer it either! As always, please feel free to reach out if you find any part of the assignment, including this section, to be unclear.

3 Problem Scope

The objective of this assignment is to finish implementing a spreadsheet program. A finished program fulfills the following functional requirements (FRs):

1. A user can navigate between cells and modify or clear the value of each.
2. When a user enters a new value, it is interpreted as either text, a number[1], or a formula.
   (a) A value which starts with an equals sign (`=`) is always interpreted as a formula; if it does not form a valid formula, then the user is shown an error.
   (b) A valid formula consists of one or more numbers or cell coordinates (e.g., `B7`) separated by addition signs (`+`).
   (c) A value which consists of one or more decimal digits (`0` through `9`) with at most one decimal point is interpreted as a number.
   (d) A value which does not start with an equals sign and is not a number is interpreted as text.
3. When a user navigates to a cell, a textual representation of its value is shown in an editable field (i.e., the content field at the top of Excel's interface), and the user can edit the value of the cell based on that representation. When the cell contains a formula, this should be a representation of the formula itself, not the computed value (which is shown in the cell itself).
4. When the value of a cell changes, the displayed contents of all formula cells which (directly or indirectly) depend on it are updated.

You are required to design and implement whatever data structures and algorithms are, in your professional opinion based on what you have learned in this course, necessary and apprioriate to fulfill the functional requirements. Additionally, your solution must adhere to the following non-functional requirements (NFRs):

1. For each algorithm implemented, it should not be possible to achieve a better time complexity (discussed in week 6 lectures) using only the tools taught in this course.
2. Any dynamically allocated memory should be freed once it is no longer needed.
3. The code must be sufficiently organized and commented for a software engineer with appropriate background knowledge to maintain.

We will ensure that the course material is scheduled such that you have adequate time to incorporate relevant concepts into your assignment (for NFR 1).

[1] Using `float` or `double` to store numeric values is acceptable.

4 Deliverables

The deliverables consist of the code for the completed program, which must be reasonably commented, and a report describing the design. The program must implement the FRs and NFRs outlined in §3. The report must include the following components:

1. A title page which includes your name, the date of submission, and the following statement:

   I, (name), attest that all of the materials which I am submitting for this assignment are my own and were written solely by me. I have cited in this report any sources, other than the class materials, which I used in creating any part of this assignment. Furthermore, this work adheres to the policy on generative artificial intelligence as documented in the instructions.

2. A brief executive summary.

3. A body with the following sections:

   (a) Design Proposal: What is your approach to solving the problem? What data structures and algorithms did you design as part of your solution, and why do you believe they are the best solution (including NFR 1)? How did you ensure that each of the FRs is satisfied? This should discuss the design, not the C-specific implementation you developed (This also allows you to receive some credit for a design that you weren't able to get fully working in the code).

   (b) Implementation: How does the code you wrote relate to the design proposed in the previous section? This should discuss each data type and function you created in the code, and include anything else you considered while writing the code (e.g, how you ensured NFR 2 was satisfied).

   (c) Testing: How did you test the program to show it meets the FRs? Include screenshots of your running program and any other relevant materials.

4. Citations for any resources, other than the course materials, which you used in creating your solution. This could include, for example, a StackOverflow answer which helped you get past a bug or compiler error, a blog post which discusses use cases for a particular data structure, or, as discussed in §2, generative AI tools used appropriately in the assignment. These don't need to be in a particular citation format; a footnote with a URL is fine. The context of the citation should make clear what information or ideas came from each source.

The code must be submitted as a ZIP file containing the entire project (i.e., including files provided with the starter code). The file name should have the format `LastName_ELEC278_Code.zip`. The report must be submitted as a PDF file. Its file name should have the format `LastName_ELEC278_Report.pdf`. These two files should both be attached to the assignment submission on OnQ. If you are unclear about any part of the submission format, please reach out on the relevant part of the OnQ discussion forum.

The assignment will be evaluated using the rubric shown on the assignment page on OnQ.

5 Starter Code

The starter code is provided in a ZIP file attached to this document. It is set up as a CMake project which should be usable from any compatible development environment, such as Visual Studio Code or CLion. It contains 9 existing files:

- `defs.h` contains some type definitions shared by the whole program.
- `interface.c` contains the code which displays and manages the text-based spreadsheet user interface.
- `interface.h` contains declarations for functions defined in `interface.c` which you may make use of.
- `model.c` contains the functions that you need to implement for the assignment; you may also put your data structure definitions and any other code you find necessary here.

- `model.h` contains the declarations for functions defined in `model.c`, so that the interface code can call them.
- `testrunner.c` contains support code for running tests to simulate user input, in case you are unable to use the interactive program (see §7).
- `testrunner.h` contains declarations for functions defined in `testrunner.c`.
- `tests.c` contains the actual tests that are run by the "testrunner" application.
- `tests.h` contains declarations for functions defined in `tests.c`.

The only file you should need to modify to complete the assignment is `model.c`.

6 Instructions

To help guide you through the process of this assignment and understand the requirements, this section provides a suggested breakdown into a series of smaller tasks. Each task includes both a design/documentation component and an application component. You do not need to follow these exact tasks in this order when writing the assignment; all submissions will be marked in accordance with the rubric, so you just need to make sure that you meet all of the requirements set out there (and above in §3).

6.1 Task 1: Basic Cell Contents Data Structure

Create a data structure to represent the contents of cells. At this point, only consider cells which are text or numbers. Some things you need to think about are:

- How are you going to organize the contents of all of the cells in a single data structure? How will you fetch or update the contents of a prticular cell given its coordinates?
- How are you going to represent the contents of each cell? How will you distinguish between numbers and text? What should the initial "blank" state of the cell be?

As part of this task, you should implement the `model_init` function and create any data types and global variables that you think are necessary for completing the task. For example, you might want to store the data structure in a global variable, and you might mant to create one or more custom data types (with `struct`, `union`, etc.) as part of defining the data structure. As mentioned earlier, all of the code you write for the assignment can go in `model.c`. Note that completing this task won't affect any of the observable behaviour of the program; that will have to wait for the next task.

In order to fulfill the deliverable requirements, it is recommended that you start by designing the data structure on paper, and then implement it in code. You may even want to start thinking a bit about task 2 as part of the design; there is no one right way to approach it. In either case, make sure that you document both what the data structure is and a summary of why you made the design decisions that you did.

6.2 Task 2: Cell Update and Display Algorithms

Create algorithms for modifying and querying the data structure you defined in task 1. You should be able to update the stored value of a cell and fetch a textual representation of a cell. Some things you need to think about are:

- When updating the contents of the data structure, how will you make sure that you correctly manage any dynamically allocated memory? When do you need to free an allocation?
- Do you ever need to duplicate data when passing it around? Pay close attention to the comments documenting the functions in `interface.h` and `model.h`.

As part of this task, you should implement the `set_cell_value`, `clear_cell`, and `get_textual_value` functions, as well as creating any additional functions that you think are necessary for completing the task. As with task 1, you will need to document the algorithm and the reasoning behind any design decisions you made in developing it.

Note that you need to produce textual representations of the value in two places: first, when the value of the cell is set in `set_cell_value`, you need to call `update_cell_display` with a textual representation to display in the cell itself. Additionally, `get_textual_value` needs to return a textual representation for the top bar, for the purpose of editing the contents of a cell. At this point, these two textual representations can be the same, but that will change in task 4.

6.3 Task 3: Make the Data Structure Support Formulae

Here, you will update the design of the data structure you created in task 1 to incorporate formulae. Remember that the only formulae you need to support consist of one or more cells or constants added together (e.g., `=A2+B2+0.4`). You will need to figure out how to store formulae in a structured form, so that they can be evaluated (e.g., so that the contents of cells `A2` and `B2` can be looked up). You also need to make sure that it is possible to reconstruct the formula, as it was originally entered, for editing purposes in the `get_textual_value` function.

6.4 Task 4: Make the Algorithms Support Formulae

Now you need to update the algorithms implemented in Task 2 to support entering formulae and generating their textual representation. Here, the difference between the two kinds of textual representations arises: the textual representation passed to `update_cell_display`, used to display the contents of a cell, should contain the computed value of a formula, while the textual representation returned from `get_textual_value` should contain the representaion of the formula itself (i.e., in the same format that it was originally entered in).

6.5 Task 5: Make Dependent Formula Cells Update

In a spreadsheet application, when you update the contents of a cell, the displayed value of any other cells which directly or indirectly depend on it are updated as well. Creating a solution to this problem is an interesting task, and there are many possible ways of approaching it. This may involve additions to both the data structure and the algorithms. There are two error cases which you should think about:

- What happens if there is a circular dependency between cells? For example, `A2` contains a formula which depends on `A3`, which itself contains a formula which depends on `A2`.
- What happens if one of the cells a formula depends on does not contain a number?

In either case, there are many different reasonable ways of gracefully handling the error, but the key point is to ensure that the program does not crash or take unexpected actions.

6.6 Task 6: Quality Assurance

It's now time to show that your program works as intended! Plan a testing process which shows that all of the features you implement work, in accordance with the requirements in the rubric. Your documentation should include both instructions for testing (e.g., "type '5.3' into cell 'A4', ...") and screenshots or other proof of the results.

7 Troubleshooting

The program uses a special library called `ncurses`, which is provided by your operating system (or by the `mingw` package, if you are on Windows), in order to show the simple user interface in the console. This doesn't affect any of the code you need to write, as it is only used in `interface.c`; however, we have noticed some issues with support for this in some cases. In particular:

- The built-in console in CLion does not fully support this. On Windows, see these instructions to install the new Windows Terminal app from Microsoft and set it as default, and then in CLion, click the three dots beside the run button, chose "Edit", enable the "Run in external console" option, then click "Ok"; now, when you run or debug the program, it should open it in the new Windows Terminal app, which fully supports these features.

  If you are on macOS or Linux, that option is not available in CLion, so your best bet to run the program is to manually open a terminal and run it there. On macOS, follow these instructions to open a terminal window in the project folder (i.e., the one which contains the files listed in §5). Then, to run the program after making changes, first compile it by pressing the "Build" button (hammer icon) in CLion, then switch to the terminal and run the command `cmake-build-debug/interactive`. Unfortunately, there is no easy way to use CLion's debugger with this method; see the last bullet point for an approach for that.

- VS Code's built-in console does work, but I've noticed that the cursor sometimes looks glitchy on Windows (i.e., appears one character left or right of where it should). That is fine and doesn't affect the actual behaviour of the program.

- On any platform, if you find that the console that opens is not wide enough to show the whole spreadsheet, this will cause issues with the user interface. You may decrease `NUM_COLS` in `defs.h` to reduce the number of columns to make it fit; this will not result in any loss of marks.

- If you are unable to use the above solutions to run the program, you can run the "testrunner" program instead of the "interactive" program. This manually simulates a series of edits to the spreadsheet, as if a user had typed them, and reports an error if unexpected behaviour is observed. Since this is not interactive, it does not use `ncurses`, and so you should be able to run or debug it on any platform without issue. The code that generates these edits is located in `tests.c`; feel free to edit this if you want to try to make it test something different.