CS130 Project 1 - Design Document
==========

Please answer all questions in this desig n document. Note that the final feedback section is optional, and you are not required to answer it if you don't want to.

Unanswered or incompletely answered questions, or answers that don't actually match the code/repository, will result in deductions.

Answers don't have to be deeply detailed! We are mainly looking for an overview or summary description of how your project works, and your team's experiences working on this project.

Logistics (7 pts)

L1. [2pts] Enumerate all teammates here.

Kevin Do, Eli Kugelsky, Zack Dugue

L2. [2pts] What did each teammate focus on during this project?

Zack - Lark parser, Workbook design
Eli - Workbook, GitHub, toposort, Workbook Design
Kevin - Tests, Workbook, toposort, Workbook Design

L3. [3pts] Approximately how many hours did each teammate spend on the project?

~10 hours each

Spreadsheet Engine Design (20 pts)

D1. [3pts] Briefly describe the high-level design abstractions (e.g. classes and/or submodules) in your spreadsheet engine, and what purposes these various abstractions fulfill.

- Class Workbook:
    - Contains all the sheets for a given workbook
    - Contains a dictionary of all the dependencies, could be a child or parent dependancy does not matter.
    - Updates the workbook when a formula is parsed or a cell that is being relied on changes

    - Has Formula parsers
- Class Sheets:
    - Here all the cells are stored in each sheets data dictionary
    - We also store the extent
    - The sole purpose of this is so that we can save the extents using a min heap for faster extent times
- Class Cell:
    - We store a list of the cell's parents and cell's children
    - We store the content and value for each cell within the cell
    - We also store the cells location
- Parser:
    - Takes care of all the formula parsing
- Class CellError:
    - A cell error class which relies on CellErrorTypes

D2. [4pts] Why did you choose the design you chose? Describe other designs (or perhaps earlier versions of the above design) that you considered, and why you ended up not using them.

We chose this design after much trial and tribulations. Initially we had the same workbook setup but each sheet would be a 2d array. We also had an individual class for each cell type we'd expect to encounter. We decided to forgo both of these idea after a discussion with Professor Pinkston after class one day. He mentioned that having a sparse representation for cells within a sheet would be a lot more efficient. Celltypes were also deemed a unnecessary encapsulation because we would really only dealing with 3 cell types. We also discussed how to store all the dependencies. Many ideas were floated around for that idea, but it seemed that many of those ideas were really the same just said in a different way. We ultimately decided to store each cell's dependencies within the cell itself but we also store any cell where it is a dependency or has dependencies within a large bag of dependencies.

D3. [4pts] At a high level (e.g. pseudocode or higher), enumerate the steps that your spreadsheet engine goes through when a caller sets a cell's contents.

First we make sure cell is within a valid sheet. We also check that the cell name given is valid, as it must be within a given range. Then we set the cell's content instance attribute to the content given as well as set the value for the cell. This checks to see if we have a string, formula, or a decimal. If its a string we just set it as so, if its a formula we wait until it gets parsed then set it to the parsed value afterwards, if its a decimal we check to see if its finite, if so we then remove exponent and floating zeros. Otherwise we treat it as a string.

D4. [3pts] How does your spreadsheet engine identify what other cells need updating when a cell's contents are changed, and what order to update them?

We have a bag of dependencies which is stored in the workbook class. So this means that if a cell is a formula cell and contains “children” cells whose value it relies on, we add the formula cell to our bag of dependencies. Each cell stores its own children as well as its parents, or other cells which relies on the given cell for its value. Thus, when we update a cell we first check if this cell is within this bag of dependencies. If not, do nothing in relation to updating other cells. But if its is within the bag of dependencies we then jump down the rabbit hole.

Rabbit Hole:

First we check if the cell is a child of another cell but is not actually placed within is supposed sheet(aka it was referenced but had None value initially), we drop it in to the respective sheet if so. Next we set the cell’s contents.

Then if the cell is a formula, we parse the formula for a value. Update the cells value with the correct formula value. Then check for incorrect formula, if so see next paragraph. We then add all of the children cells to the current formula cell. This means that for every cell referenced within the formula string we add it to the cell.children list. At the same time we are doing this we also place every cell being referenced inside our bag of dependencies. If a cell being referenced does not exist we create a None cell. These are cells that are in our dependency list, but not in our actual sheets list. For each child cell we also add the initial formula cell to the children cells.parent list. This allows us to have edges which are double linked between cells references.

After further review, we don’t actually need children's cells as we only need to propagate things to their parents. This will likely be removed by the next project.

If however the cell is deemed to be an incorrect formula (ex. Incorrect reference) we then propagate back up to its parents to set them all to the CellErrorType corresponding to the error (I’ll discuss how we propagate soon). The cell’s value will get set to the error as well during this propagation.

Recompute the parents and Children (Propagation):

We run a topological sort on the cells parents first. This means we visit each of the cells parents, and if those cells have their own parents we visit them too. This allows us to store each parent cell we need to update given the change in the current cell. We store all the captured the parents, grandparents so on and so forth in a list. For each of these parent cells, ordered topolgically, we recompute the cell value by parsing the formula and setting the formula value to include the now new updated cell. We assume rightfully that all parent cells will be formulas.

That is all.

D5. [3pts] How does your spreadsheet engine identify cycles between cells when a cell update is performed? Are cells in a cycle processed any differently from other cells outside of the cycle?

During our cell update when the topological sort is called, we add every cell seen to a ‘cycle_check’ set. If a later cell, which would be a cell's parent, already exists in the cycle_check, then we know that the parent cell has already been seen in a prior iteration as a child cell itself and there for a cycle must exist. Let me give you an example. Let’s say we have two cells which reference each other A1 <-> B1. When we visit A1 initially, we add it to our cycle_check. Then we visit add all of A1’s parents to be visited, which in this case is B1. So now we are at B1. We also add B1 to the cycle check. Now we add all of B1’s parents to be visited,

this being A1. However, we now notice that A1 was already visited before and thus we have found ourselves a cycle.

When the cycle is detected we set the cell as well as the parent cell to a CellErrorType of CIRCULAR_REFERENCE. Note that since the cell is never visited, we will dfs to set all the parents in the cycle_check to the the CellError CIRCULAR REFERENCE. Thus when we run the lark parser afterwards, during our recomputing of the formula value, we will be hit with a CIRCULAR_REFERENCE and thus all cells in the formulas / references/ dependencies affected will be set, as well, to CIRCULAR_REFERENCE.

D6. [3pts] What steps does your spreadsheet engine go through when a sheet is deleted from a workbook? How does it identify cells that may need to be recomputed after a sheet-deletion operation?

When a sheet is deleted from the workbook we first make sure the sheet is a valid one. After that we check to see if there were any cells within the given sheet that have parent dependencies. This would mean, do we have any cells outside of the given sheet which reference any cells within the given sheet. For those cells which are being referenced, we call our function which recomputes the cell's value, which is now set to none, and then propagates the new value to all of its parent cells. This function then runs a topological sort separately for all cells within the sheet and updates all of the parents to include the none value. We then delete the sheet from our workbook's dictionary of sheets.

When we add a sheet back in, if the name matches any of the referenced cells, we go ahead and update all the cells.

Implementation Process (23 pts)

P1. [4pts] How did your team break down and keep track of the various tasks to complete for this project? Did you use the GitHub issue tracker, or some other tool like Trello?

We first wrote a google document on the things we needed to tackle. After that we sort of just went at the code and over time we'd check the project requirements then update the project per the requirements. We are now starting to use the issue tracker on Github for details/ requirements that we are missing or bugs we are noticing. Our messenger group chat also has a lot of the problems and stuff that we were facing.

P2. [4pts] How did you assign tasks to teammates? Did you stick with your task-assignments through the project, or did you shift around tasks as the project progressed? Why?

We didn't do a formal task assignment. We saw fit that since we could all meet relatively often and were all able to code together during these times, we would do so. At least for the beginning of the project we wanted all of us to have a good understanding of codebase during its development. At middle of the project we sort of split into two, one where Kevin and Eli

focused on the Workbook implementation as well as all of the Cell and Sheet classes and Zack then took lead of the parser implementation and how we would approach that. Every design implementation detail was gone run over all 3 of us so that we all know how things work.

I think for the next project we'll be much better at deliberately breaking up the tasks to individuals.

P3. [4pts] How would you characterize the quality of your project testing? Make sure to touch on these items: Is it automated? Is it easy for teammates to run? Is it fast? Is it reasonably complete?

We were late to the testing party. The majority of our tests we're implemented after a good chunk of our features were already inplace. We were not able to test in the beginning because we didn't really know how to A) construct those test, B) what exactly to test for. Now we do know what to test for and have a test for almost every requirement laid out in the project specification. The tests are mainly automated end-to-end tests which create a workbook and follow some design characteristic we need to have working. We have some unit tests that check basic functionality like setting a content, adding and deleting a sheet. The tests are run with one line, although we will be working on this to add it to the readme as well as better documentation on the testing. It is fast and to our belief complete.

P4. [3pts] What mechanisms did your team use for communication during the project? Did you find them to be effective? What issues did you encounter, if any?

We used a groupchat with all of us in it. We had basically zero communication problems and we worked as a very well oiled machine. We liked that all of us could whos planning on working when/with who so another can join if they have the time. If one person worked on something solo, they would make a new branch off of whatever branch we were working on and we would create a pull request so that all the members could see and approve the changes, or we would just approve them when we met up again.

P5. [3pts] Did you use any kind of collaborative construction techniques, e.g. code reviews or pair-programming, during the project? If so, what are your observations about its usefulness? Did you find it to have any downsides?

We did pair programming using liveshare in VS code. It was very good to work directly next to a fellow programmer on a problem. We were able to either tackle a problem together head on or one tackles the head while the other tackles the tail or the issue/ feature. Also debugging while pair programming was extremely efficient. We also got together as a group and reviewed pull requests.

P6. [5pts] What would you like to improve about your team's development process, going forward?

I think we just need to lay out the goals, write the tests, and update the issues tracker as soon as a bug is found and squashed. The test should be our benchmark if a feature was added

correctly. We also need to keep better documentation/ typing/ docstrings so we aren't doing that last minute.

Section F: CS130 Project 1 Feedback [OPTIONAL]

These questions are OPTIONAL, and you do not need to answer them. Your grade will not be affected by answering or not answering them. Also, your grade will not be affected by negative feedback - we want to know what went poorly so that we can improve future versions of the course.

F1. What parts of the assignment did you find highly enjoyable? Conversely, what parts of the assignment did you find unenjoyable?

F2. What parts of the assignment helped you learn more about software engineering best-practices, or other useful development skills? What parts were not helpful in learning these skills?

F3. Were there any parts of the assignment that seemed _unnecessarily_ tedious? (Some parts of software development are always tedious, of course.)

F4. Do you have any feedback and/or constructive criticism about how this project can be made better in future iterations of CS130?

I think that it would be cool to have a demo of the solution code to see what's happening. A lot of the functionalities don't look the same as microsoft excel and stuff like that.