CS130 Project 4 - Design Document
==================================

Please answer all questions in this design 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, Eli, Zack, and Purvi

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

Kevin - Functions and Tarjans
Zack - Boolean and Functions
Eli - Tarjans and Functions
Purvi - Tarjans and Testing

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

13 hours

Spreadsheet Engine Design (31 pts)

D1. [3pts] Briefly describe the changes you made to the Lark parser grammar
to support Boolean literals.

We added a ```BOOLEAN -> boolean``` to the base nonterminal. Where BOOLEAN catches case-insensitive “TRUE” or “FALSE” strings. So it’s just treated as another base type like into or string.

D2. [4pts] Briefly describe the changes you made to the Lark parser grammar
to support conditional expressions. How did you ensure that conditional

operations are lower precedence than arithmetic and string concatenation

We added conditional expressions as operators. In the formulas.lark file, we basically copied the structure of the lark grammar used for add_expr. The comp_expr uses a COMP_OP which catches all comparison operations like “<” or “=”. We also had to replace the top-level expression nonterminal so that it is a comp_expr instead of an add_expr | concat_expr. Overall it was a very simple implementation.

D3. [6pts] Briefly describe how function invocation works in your spreadsheet engine. How easy or hard would it be for you to add new functions to your engine? What about a third-party developer? How well does your code follow the Open/Closed Principle?

Functions are stored in a dictionary in the workbook with keys as function names which are grabbed by the parser. It is very easy to extend this because you just update the dictionary with the new function, but it is closed to modification because the user can’t delete old functions! Our can also be easily adjusted to handle the case where custom functions error out. As a proof of concept for this we even included an extra function SUM() which helped for some of our testing purposes!

D4. [4pts] Is your implementation able to lazily evaluate the arguments to functions like IF(), CHOOSE() and IFERROR()? (Recall from the Project 4 spec that your spreadsheet engine should not report cycles in cases where an argument to these functions does not need to be evaluated.) If so, what changes to your design were required to achieve this? If not, what prevented your team from implementing this?

Yes, our implementation lazily evaluates the aforementioned arguments since we only evaluate the parts we need by selectively evaluating arguments in our function implementation based on the logic of the function. This complicates maintaining our graph for cycle reporting though. The way we addressed this was by actually collecting cell references as we evaluate the formula. IE we use the “cell” argument in our interpret to append to a set of cell refs, and that way exclusively the cells that are actually evaluated are the ones that end up in our dependency graph.

D5. [4pts] Is your implementation able to evaluate the ISERROR() function correctly, with respect to circular-reference errors? (Recall from the Project 4 spec that ISERROR() behaves differently when part of a cycle, vs. being outside the cycle and referencing some cell in the cycle.) If so, what changes to your design were required to achieve this? If not, what prevented your team from implementing this?

We use Tarjans. There were a few things that we had to update. Firstly, we changed our implementation to prune parents when they are unnecessary. Before, we just had unnecessary recomputes, but now all the recomputes are right as we prune our parents. From there, we use tarjan's algorithm to detect the strongly connected components. Therefore, when we do the ISERROR function, the error doesn't propagate as it's not in the strongly connected component. If it's not in the strongly connected component but has a cell reference in the strongly connected component, it just evaluates to true.

D6. [4pts] Is your implementation able to successfully identify cycles that are not evident from static analysis of formulas containing INDIRECT()? If so, what changes to your design were required, if any, to achieve this? If not, what prevented your team from implementing this?

Our code somewhat works with this. Basically what's wrong is that we are doing a topological sort with tarjans, but because the graph changes during the evaluation, we can't really get it to redo the topological sort as a result of the indirect. We were trying our best to call recompute cells and parents, but we believe that it is getting overwritten regardless of what we're doing, but we'll probably figure it out by next project. Besides this pretty complicated edge case with this, our code finds some stuff out.

D7. [6pts] Project 4 has a number of small but important operations to implement. Comparison operations include a number of comparison and type conversion rules. Different functions may require specific numbers and types of arguments. How did your team structure the implementation of these operations? How did your approach affect the reusability and testability of these operations?

To be honest I think a lot of our comparison rule stuff was somewhat adhoc. We created a special function to handle conversions between different types, and another function to handle converting strings, decimals, etc. into bools. We wrote very extensive tests in order to ensure that we were performing proper conversions for comparison operations, so we are relatively confident our implementation is correct.

Performance Analysis (12 pts)

In this project you must measure and analyze the performance of features that generate large bulk changes to a workbook: loading a workbook, copying or renaming a sheet, and moving or copying an area of cells. Construct some performance tests to exercise these aspects of your engine, and use a profiler to identify where your program is spending the bulk of its time.

A1. [4pts] Briefly enumerate the performance tests you created to exercise your implementation.

We created a load workbook profiler test that loads the workbook repeatedly from a pre-saved workbook. Next, we created a copy sheet test that copies a given sheet multiple times. We have a rename sheet test that creates new sheets and renames them multiple times. We then have 2 copy and 2 move cells tests. The first of the copy and move cell tests run the copy or move functions multiple times. In the second tests, we run the copy or move cells function once each. However, we copy or move a very large number of cells.

A2. [2pts] What profiler did you choose to run your performance tests with?
Why? Give an example of how to invoke one of your tests with the profiler.

We ran the performance tests with Cprofiler because it is built in, simple to use, and measures the time spent at every function, which is convenient because we want to check the time spent at each one of our functions (load workbook, copy cells, etc) as opposed to seeing the time spent in each of the lines. We invoke the tests with the profiler by accessing the test directory and running the profiler.py file. All of the profiler logs get added to the profile_logs directory.

A3. [6pts] What are ~3 of the most significant hot-spots you identified in your performance testing? Did you expect these hot-spots, or were they surprising to you?

Pro_write_formula_chain, pro_cycle_break_unbreak, and pro_set_cell_contents overall take the most amount of time. Within the pro_cycle_break_unbreak function, we spend the most amount of time in the Graph.py file, and specifically, in Tarjan's function. This is expected because Tarjan's algorithm can take a lot of time, especially with a larger graph. Our implementation also has some inefficiencies like looping through the parents for each cell twice in some cases. In the pro_write_formula_chain function, we spend a lot of time in the parser, and specifically in the earley.py, earley_common.py and earley_forest.py functions. Since these are in the Lark Parser, we know that is another hotspot. This is expected because EARLYE is a slow parser, as opposed to other parsers like LALR. Lastly, we spend a lot of time in Recompute cell value because of the way our recompute cell algorithm is structured we needlessly recompute values (this is due to us having to handle the graph structure changing with evaluation).

Section F: CS130 Project 3 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?

We really liked working on Tarjans. It was super interesting to have to take it from a recursive algorithm into an iterative algorithm. We did this by literally creating our own function call stack and giving priority to calls that are newer on the stack. We had to split up the recursive aspect into two different loops. One where the cell has not been seen before, and another when after the “recursive call” it has then been seen. The least enjoyable part for this assignment was probably dealing with the numerous edge cases that can exist with function calls in excel.

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?

The most helpful part of this assignment hands down has to be the fact that we need to jump head first into our code. AKA we became much more agile in the way we developed compared to prior projects because we realized there would be loads of edge cases that would only surface later on.

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

Trying to figure out INDIRECT has to be the most tedious part of this assignment. It has been extremely challenging to update if we had a cellError.

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

Not really. It was a good challenge. We got everything we believe we could accomplish done and just barely got enough of the INDIRECT completed. It works as intended if you don’t stress it so much.