**Report**

**User Manual:**

* A menu bar is present on the top of the window.
* The menu has an option to close the windows.
* Open
* The spreadsheet is an 8x8
* Click on a cell
* Enter a valid expression into the cell.
* A valid expression contains: A-Z, constants, and operations. Supported operations are addition, subtraction, multiplication, and division.
* A valid expression would be something like “A1+2”.
* To retrieve a value from a cell, call the cell location, for example “B2”.
* To change the value in the cell, click the cell location and press backspace to clear.
* Press Enter to calculate the expression inside the cell or to assign a value into it.
* The cells after column Z will contain double characters representing AA and etc.
* To write an expression inside the parenthesis, make sure it is valid.
* A valid parenthesis will look like (A1+2)+3.
* The parenthesis ensures the program calculates the expression in PEMDAS.
* To exit the program, click on the close button on the top right of the screen.

**Technical Manual**

Data structures used:

Linked-List: We used the linked lists to represent the dependencies of each node. All the edges from a node to another node were stored in a linked list.

HashMap: The hashmap stored a cell as its key and the value was a linked list of every dependency that cell had. It was used in the topological sort algorithm to get all the edges of a node.

Stack: We implemented the stack data structure from Weiss’s code. The stack is storing the tokens so that when it is read from the bottom up, it will return in a postfix expression. The tokens that the stack is storing depends on what the user has entered into the spreadsheet.

Queue: The queue was used for the topological sorting algorithm. All elements with indegree equaling 0 were put into the queue. After that, they were popped out one by one and then Kahn’s algorithm was performed on it.

Tree: Binary tree was used for the expression tree class which is one of the fields of the cell class. Expression tree class keeps all different tokens represented by the expression tree node in order in which operator token (node) will be the parent of either literal token (node) or cell token(node). Operator token (node) must have both children. Literal token and cell token are leaves of expression tree. The expression tree was also used to evaluate the value of cells.

**Algorithms:**

Kahn’s algorithm:

This algorithm is a version of topological sort. It works by finding vertices that don't have indegrees and removing the outgoing edges from the vertices. There will be at least one vertex that doesn't have indegrees.

Depth-first search algorithm:

The depth-first search algorithm traverses the tree. It starts from an arbitrary node and moves to the adjacent nodes until there it can't move to another node. It then backtracks and does the same thing again.

**Group members worked on:**

Trevor: I implemented the Spreadsheet class which contains a two dimensional array of cells representing the grid. Inside the class, a graph is created and topological sort is implemented using Kahn’s algorithm by finding each node that has an indegree of zero and removing it from the graph and all edges to it. If the graph contains any edges after the algorithm, then we know there is a cycle in the graph. If there is a cycle, the program informs the user and reverts the value back to the previous value and formula. The spreadsheet class also includes some methods for printing values and formulas which are used in the SpreadsheetApp class. For the GUI, I used a JPanel on top of a JFrame which contains a JTable of the values. The table’s column headers are populated using the style of excel with letters for the values. The row headers have integer values from 0 to the number of columns like excel. There is an action listener on the table that detects when the user changes a value in the table. When it is called, the spreadsheet does topological sort and evaluates the values of the spreadsheet and displays them in the table.

Phuoc: I worked on the Cell, ExpressionTree, ExpressionTreeNode. The cell class contains formula, value, expression tree. When cell class is constructed with a formula that it builds the expression tree from a given formula. There is an evaluate method which calculates the value of the cell based on the expression tree which is built when cell class is constructed. Evaluate method is made by recrusively checking if node has operator token, we add its children value together. If node has literal token, we return its value. If node the has a cell token, we return its value by using row and column in cell token as indexes for the spreadsheet.

Aman: I worked on the CellToken, OperatorToken, and LiteralToken classes. We also had multiple errors pop up relating to the different classes. The cell token class had two methods which were the getCellToken and printCellToken. This got the coordinates of the cell token. I also needed to add the setters and getters for that class. The OperatorToken class also had three methods which were isOperator, operatorPriority and priority. I had to add the constructor for the token and the getter method. The LiteralToken class was only getting a literal token that stored the constant. We also had errors that we needed to fix with getFormula because it had typos and we weren’t able to use it within the expression tree class. We implemented code into the token class to be able to call it by making it static. The cell class had an issue with the evaluate method so we had a helper method that performed the different operations on the cell. It returned the different token depending on what type it was. We also had an issue with the spreadsheet which resulted in more columns and wouldn’t evaluate the expression. I just had to change the col+1 to col and j to j-1 for it to work.

Bohdan: I created the Save Spreadsheet to a file button and its functionality. I made a filter that would only allow user to open a text file. The function saves the table in a file where values in each row are separated by commas and then each new row is a new line. I also attempted to make the open from file functionality, but then it didn’t work correctly.