**Assumptions Made**

When the columns letters exceed the number of letters in the alphabet the next value would index at the start of the alphabet and combine. i.e., a1 ->b1 -> … z1 -> aa1 -> ab1 -> … az1

**Trade-offs / Design Decisions**

The use of a HashMap to store the results could be considered a poor decision of data structure, as I can’t modify the values while iterating, this could have reduced the number of loops over the not seen list.

**Code Structure**

* Create a Queue to store operands
* Create a Map to store our results key = location (column, row), value = result
* Create a Map of not seen values
* Scan a cell from left to right and follow the below statements for every scanned element in the cell
  + If the element is a number, push it on the queue
    - If the element is an operator, pop the element from the queue and evaluate the operator, push the result onto the queue
    - When the expression has finished and there are no more elements in the cell and queues size equals 1 then that is our final answer for that cell
  + If the element or cell is a “lookup operation” i.e., “a1” or “a1 b1 / c1”
    - Lookup the value based on it’s key (location) in the results map if we have a match then push it on the queue else store it in a not seen map and push error to the results map as we have yet to see that location.
    - Proceed to follow previous steps of adding to a queue and waiting for an operand.

*The results map now contains a combination of solved postfix expressions, regular single values, and error values as we are yet to see a reference to a location ahead of the list. This is the process to do a second to ..n amount of passes to solve the missing references.*

* For each element in the not seen map
  + If the element is a number, push it on the queue
    - If the element is an operator, pop the element from the queue and evaluate the operator, push the result onto the queue
    - When the expression has finished and there are no more elements in the cell and queues size equals 1 then that is our final answer for that cell
  + If the element or cell is lookup operation i.e., “a1” or “a1 b1 / c1”
    - And the cell doesn’t equal to the error value push the value onto the queue otherwise break as it has still yet to see this value.
    - Proceed to follow above steps of adding to a queue and watching for an operand
    - Add the result to the results map and remove from the not seen map
* If the not seen map is not empty, it means there is a genuine error with the input as it should have seen every element by now – write an error to the results map.

*At this point the results map will have either computed values or error messages, the result now be written to STDOUT.*