Programming Question – Spreadsheet Calculator

A spreadsheet consists of a two-dimensional array of cells, labeled A1, A2, etc. Rows are identified using letters, columns by numbers. Each cell contains either an integer (its value) or an expression. Expressions contain integers, cell references, and the operators '+', '-', '*', '/' with the usual rules of evaluation – note that the input is RPN and should be evaluated in stack order.

Write a program (in **Java or Scala**) to read a spreadsheet from 'stdin', evaluate the values of all the cells, and write the output to 'stdout'.

The spreadsheet input is defined as follows:

- Line 1: two integers, defining the width and height of the spreadsheet (n, m)
- n*m lines each containing an expression which is the value of the corresponding cell (cells enumerated in the order A1, A2, A<n>, B1, ...)

Your program must output its data in the same format, but each cell should be reduced to a single floating-point value. For example, we would expect the following expect to produce the indicated output:

Input	Expected Output
3 2	3 2
A2	20.00000
4 5 *	20.00000
A1	20.00000
A1 B2 / 2 +	8.66667
3	3.00000
39 B1 B2 * /	1.50000

The above example input visually looks like:

```
  | 1            | 2     | 3            |
--+--------------+-------+--------------+
A | A2           | 4 5 * | A1           |
--+--------------+-------+--------------+
B | A1 B2 / 2 +  | 3     | 39 B1 B2 * / |
-------------------------+--------------+
```

Comments:

- Scala programs should compile with Scala 2.10.5 or 2.11.X.
    - Scala solutions should only use Scala or Java standard libraries (unit testing libs are OK).
    - Scala solutions should define their main method/Application object in a file named Spreadsheet.scala
    - We will use the following command to build your code: "scalac *.scala". If you require build options more sophisticated than that, you must include a bare SBT build definition file.
    - We will run your code like this: “cat spreadsheet.txt | scala Spreadsheet”
- Java programs should compile using the standard Oracle JDK, version 1.7 or 1.8.
    - Java solutions should only use the standard Java libraries (unit testing libs are OK).
    - Java solutions should define their main method in a class named Spreadsheet.java
    - We will use the following command to build your code: “‘javac *.java”. If you require build options more sophisticated than that, you must include an Ant build.xml file.
    - We will run your code like this: “cat spreadsheet.txt | java Spreadsheet”

- Your program should detect cyclic dependencies in the input data, report these in a sensible manner, and exit with a non-zero exit code.

- All numbers in the input are positive integers, but internal calculations and output should be in double precision floating point.

- You can assume that there are no more than 26 rows (A-Z) in the spreadsheet; however there can be any number of columns (1-n).

- We will evaluate output automatically, so formatting it correctly is very important. There should be one output per line, lines organized in this order: A1, A2, ... A<n>, B1, B2, ... B<n> ... When printing out the numbers, please use formatting as below:
    - In Scala, use (“%.5f”).format(val)
    - In Java, use String.format(“%.5f”, val)

- If this all seems to be a bit too easy, further credit will be given for:
    - Extending the expression grammar to support negative numbers as inputs.
    - Extending the expression grammar to include an increment or decrement operator (++ or --)