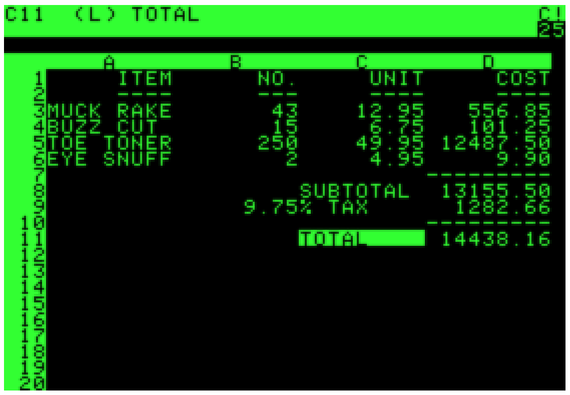
AP CS Project 2: TextExcel

Introduction

VisiCalc, created in 1979 for the Apple II, was the first spreadsheet program for the personal computer. It was also one of the first applications to make personal computers useful, and is the predecessor of Microsoft Excel, which you are more likely to have seen today. A spreadsheet is a program that lets you view, organize, and manipulate rows and columns of data in a grid. For Project 2, we'll implement our own spreadsheet, TextExcel, which will be similar to the original VisiCalc.



```
C11  (L) TOTAL                                   C!
                                                 25
          A           B          C          D
 1       ITEM        NO.       UNIT       COST
 2       ----        ---       ----       ----
 3MUCK  RAKE          43      12.95     556.85
 4BUZZ  CUT           15       6.75     101.25
 5TOE  TONER         250      49.95   12487.50
 6EYE  SNUFF           2       4.95       9.90
 7                                    --------
 8                   SUBTOTAL         13155.50
 9               9.75% TAX             1282.66
10                                    --------
11                   TOTAL            14438.16
12
13
14
15
16
17
18
19
20
```

Figure 1 - Screenshot of VisiCalc

This document describes the entire project. The Overview section explains what the finished project should look like. The sections for Checkpoints 1 through 5 below that show how you'll gradually build up from a simple program that displays an empty grid to a powerful spreadsheet. Each checkpoint builds on the previous ones. This is a substantially more difficult project than Fractional Calculator, so you'll need to be careful to keep on track.

Overview

Your TextExcel program will organize data into a grid measuring 10 rows by 7 columns. The rows are numbered 1 through 10, and the columns A through G, making A1 the top left cell in your grid and G10 the bottom right. Each cell is initially empty, but can be given a value using a command like "C4 = 3.14159".

Basic cells will contain real numbers, strings, or dates. Later, you'll build formula cells, which can contain mathematical expressions, references to other cells, and special commands to find sums and averages. In addition to setting cell values, you'll write code to clear cells, to sort them, and to print the whole grid.

On the next page, you can see a sample run of a completed program that shows a few of the available commands. You can see that it keeps prompting the user to enter a command until the user types 'quit'. As the user sets values, they are stored by the program. When the user types 'print', the whole 10x7 grid is printed.

```
Welcome to TextExcel!
Enter a command: A3 = "hello"
Enter a command: B7 = 02/06/1976
Enter a command: C1 = 3.14159
Enter a command: D8 = (2 * 3 + 5)
Enter a command: print
+------------+------------+------------+------------+------------+------------+------------+------------+
|            |     A      |     B      |     C      |     D      |     E      |     F      |     G      |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     1      |            |            |  3.141590  |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     2      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     3      |   hello    |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     4      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     5      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     6      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     7      |            | 02/06/1976 |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     8      |            |            |            |    11.0    |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     9      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     10     |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
Enter a command: D8
D8 = (2 * 3 + 5)
Enter a command: A3
A3 = "hello"
Enter a command: clear A3
Enter a command: A3
A3 = <empty>
Enter a command: quit
Farewell!
```

Checkpoint 1

In this first checkpoint, you'll get the basic structure of your program in place. You'll write code that can display the basic 10x7 grid of empty cells when the user types 'print', and exit when the user types 'quit'. To do this, you'll create three classes: one to process user input, one to contain the rows and columns of data, and one to represent an individual cell in that grid.

Your first class should be named TextExcel and should contain your program's main method. In main, you need to write a loop that will repeatedly ask the user to enter commands, accept and process each command, and keep doing that until the user types 'quit'. When the user types commands other than 'quit', you can use if statements to figure out what the user is asking for.

Hint: Write your TextExcel class first and get the loop working before you try to add other classes. Have it exit if the user types 'quit' and (for now) print out something like "Invalid command" if the user types anything else.

Your second class represents an individual cell inside the spreadsheet, so Cell would be a great name for it. For this checkpoint, your cells are basically placeholders. When the whole spreadsheet is printed, the Spreadsheet object should ask each cell for what it should print by calling a method in Cell. getValue would be a good name for this method. You don't need to store any real data in the cells for Checkpoint 1.

Your third class represents the grid or spreadsheet, so call it something like Spreadsheet. It will store a two-dimensional array of cells, so you'll need to initialize that array in your constructor. For this checkpoint, it will also need to be able to print the grid, so it should have a public method to do that.

Hint: Write a simple version of these two classes first, and make sure they don't break anything. For example, write the Spreadsheet class with an empty constructor and a print method that just prints out "I should print the grid here", and write the Cell class with an empty constructor and a getValue method that just returns a string, "value". You can come back later and fill in the constructors and make the methods work.

Hint: As you're writing the three classes above, run your program every time you change anything. Keep it in a working state, even if it's incomplete, as much as you possibly can.

Hint: Think about how you will represent empty cells in your grid. There are a couple of ways to do this, but the simplest is probably to allow your array of cells to contain nulls for empty cells. If you do that, though, be sure your printing code checks to see if a cell is null before trying to print it!

Hint: When you run TextExcel for real (or when a grader runs it), only the main method from your TextExcel class will be run. But nothing stops you from adding a main method to other classes to test them. For example, when you add your print method to Spreadsheet, also add a main method that constructs a spreadsheet and calls print. That way, you can run Spreadsheet directly and see if print is working without worrying about whether your other classes are working.

Hint: you don't have a way to add a Cell to your spreadsheet yet, so you can't see if Cells are printed correctly. But if you write some test code in a main method for Spreadsheet, you can add a cell there. Have your cell just return something like "" so you can see if it is being printed correctly.*

Checkpoint 1 is worth a total of 20 points. You will be able to find a more detailed rubric online but this is the approximate breakdown of points.

3	Functional: program compiles and runs
3	Functional: program prints the greeting and farewell messages as shown below.
5	Functional: program prints the spreadsheet when ‘print’ is typed
3	Functional: program exits when ‘quit’ is typed
3	Design: submission includes TextExcel, Spreadsheet, and Cell classes as described
3	Style: good comments, spacing, and identifier names make the code easy to read

Checkpoint 1 won’t look like much, but the code you write is the basis for the rest of your TextExcel project, so be careful to keep it well organized and neat. Here’s a sample run:

```
Enter a command: print
+------------+------------+------------+------------+------------+------------+------------+------------+
|            |     A      |     B      |     C      |     D      |     E      |     F      |     G      |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     1      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     2      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     3      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     4      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     5      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     6      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     7      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     8      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     9      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     10     |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
Enter a command: quit
Farewell!
```

Checkpoint 2

The second checkpoint is where you'll add code to allow a user to set text, number, and date values in cells. Users will be able to view cells by typing references to them, like "C3". They will also be able to set cell values by typing references with assignment statements, as you see here:

```
Enter a command: A4 = "Hello!"
Enter a command: B8 = 02/06/1976
Enter a command: D3 = 3.14
```

When the user types only a cell reference, if the cell has a value stored in it, you must print **exactly** what the user originally entered for that cell. If the user types a reference to a cell that has no value stored in it yet, you should print "<empty>" as the value.

When the user types 'print', you must print the entire grid, including the values for any cells that have values set. Cell values must be centered in the 12-character-wide cells. If a cell value is longer than 12 characters, you must print the first 11 characters followed by a right angle bracket (">"). Here are a few examples:

```
Enter a command: A4 = "Hello!"
Enter a command: B8 = 02/06/1976
Enter a command: D3 = 3.14
Enter a command: C7 = "a string that is too long"
Enter a command: B8
B8 = 02/06/1976
Enter a command: D3
D3 = 3.14
Enter a command: C7
C7 = "a string that is too long"
Enter a command: F5
F5 = <empty>
Enter a command: print
+------------+------------+------------+------------+------------+------------+------------+------------+
|            |     A      |     B      |     C      |     D      |     E      |     F      |     G      |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     1      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     2      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     3      |            |            |            |    3.14    |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     4      |   Hello!   |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     5      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     6      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     7      |            |            |a string th>|            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     8      |            | 02/06/1976 |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     9      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     10     |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
```

When the user attempts to set a value in a cell, you will have to determine what kind of value it is based on these rules:

- Strings will always have quotes around them (C3 = “I’m a string”)
- Numbers will only contain the dash, period, and numeric characters (like C4 = 35 or C5 = -4.2).
- Dates will always be in the form month/day/year.

All the inputs that will be tested against your program will follow one of those formats, so if your program crashes when it gets bad input, you won’t lose points. (But do note that gracefully handling bad input is an extra credit opportunity.)

To complete Checkpoint 2, there are four main problems you’ll have to solve.

First, you’ll need a way to look at what the user types and understand that a command like “A1 = 5.5” means you should create a number cell with the value 5.5 and store it at location A1 in the grid. If the user just types “A1”, it means you should display exactly what he originally entered for that cell, or “<empty>” if the cell is empty.

Hint: One way to figure out if the user is trying to view or modify a cell’s data is to look to see if the first part of what the user typed, up to the first space, is a valid reference to a cell. Your Spreadsheet class is a great place to put a method that checks to see if a string like “A1” or “H93” or “Huh?” is a valid cell reference, since it knows what rows and columns it has.

Second, you’ll need a hierarchy of classes to represent different types of cells. You will need to create subclasses of your original Cell class to represent string cells, date cells, and number cells. Think about what functionality should be the same for all cells and what will be different for each type of cell. Anything that needs to be shared by all cells should be in your Cell class.

Hint: you may need to “refactor” your Cell class as part of adding this hierarchy. Refactoring code means reorganizing it to meet new requirements. Don’t be bashful about deleting lines of code that are no longer useful or moving methods to different classes. While you should definitely start Checkpoint 2 using your Checkpoint 1 code, you should expect that you’ll modify some of it.

Third, you need some code to create the appropriate type of cell (date, number, or text) based on the value the user entered. This will end up being a large enough chunk of code that you should separate it from the rest of your code by creating a new class just to create cells. You can call it CellFactory, since this pattern for creating classes that are subtypes of one parent class is called a Factory.

Fourth, you may need to update the way your Spreadsheet prints values when the user types ‘print’. You should already be calling each Cell’s getValue method, but now it will return something other than an empty string. You’ll need to truncate what it returns to 12 characters if it is longer than that, or pad it with spaces on each side if it is shorter, so that it gets centered in the square where it’s printed. You may need to refactor this code and your Cell code to have two different methods, one to get the value displayed at the command prompt and the other to get the value displayed in the grid, since they are different (e.g. the string should have quotes around it at the command prompt but not in the grid).

Be sure to try all the commands shown above, as well as the commands from Checkpoint 1, and any other inputs you think of. It is ok if your program crashes when it gets invalid input, but you will have a chance for extra credit points if you can avoid that.

Checkpoint 2 is worth a total of 40 points. You will be able to find a more detailed rubric online but this is the approximate breakdown of points.

4	Checkpoint 1 functionality must still work (exit and print)
8	Can set values for string, date, and number cells
8	Can display individual values for string, date, and number cells
8	Can print the whole grid with correct formatting
4	Cells are organized into an appropriate class hierarchy
4	Code is organized into methods and classes that are each responsible for one thing
4	Code is formatted for good readability (indentation, spacing, identifier names, etc.)

Checkpoint 3

Checkpoint 3 involves two additions to your Checkpoint 2 code. You'll need to add support for clearing cells, and you'll add a new type of cell that can contain formulas.

Clearing cells requires you to support a new command, 'clear'. The clear command may be typed with or without a cell reference, like 'clear A3' or just 'clear'. When typed with a cell reference, clear should clear the value for that one cell. When typed alone, clear should clear all values in the spreadsheet.

To handle the clear commands, you'll need to think about how to parse the user input in the code where you're already looking for print, quit, and cell operations. You'll also probably want to add some methods to your Spreadsheet class that can clear individual cells or all cells, so your input-handling code can call those methods.

```
Enter a command: B2 = 123
Enter a command: C2 = "hello"
Enter a command: D2 = 10/13/1977
Enter a command: print
+------------+------------+------------+------------+------------+------------+------------+------------+
|            |     A      |     B      |     C      |     D      |     E      |     F      |     G      |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     1      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     2      |            |    123     |   hello    | 10/13/1977 |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     3      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     4      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     5      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     6      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     7      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     8      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     9      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     10     |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
Enter a command: clear C2
Enter a command: C2
C2 = <empty>
Enter a command: clear
Enter a command: D2
D2 = <empty>
Enter a command: B2
B2 = <empty>
Enter a command: print
+------------+------------+------------+------------+------------+------------+------------+------------+
|            |     A      |     B      |     C      |     D      |     E      |     F      |     G      |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     1      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     2      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     3      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     4      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     5      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
```

Formula cells are cells that contain mathematical expressions, like 3 + 2 or 19 / 5 * 4 + 6 / 2. The expressions may include addition, subtraction, multiplication, and division. No other operators or punctuation will be used. When the user enters a formula, it will always be in parentheses, like this:

```
Enter a command: C5 = (2)
Enter a command: C6 = (3 + 5)
Enter a command: C7 = (2 * 9 - 18 / 3)
```

If the user types a cell reference to display a formula cell, he or she should see the original formula as typed by the user. If the user uses the print command to print the whole grid, the evaluated result of the expression should be displayed.

```
Enter a command: C5
C5 = (2)
Enter a command: C6
C6 = (3 + 5)
Enter a command: C7
C7 = (2 * 9 - 18 / 3)
Enter a command: print
+------------+------------+------------+------------+------------+------------+------------+------------+
|            |     A      |     B      |     C      |     D      |     E      |     F      |     G      |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     1      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     2      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     3      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     4      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     5      |            |            |     2      |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     6      |            |            |    8.0     |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     7      |            |            |    12.0    |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     8      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     9      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     10     |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
```

To support formula cells, you'll need to do two almost completely unrelated things. First, you will have to add a new class to represent formula cells, which you should be comfortable with after adding classes for string, date, and number cells in the previous checkpoint. You'll need to update your cell factory to be able to create formula cells.

The second part is trickier. You need to be able to evaluate mathematical expressions. You do NOT need to handle correct order of operations, though you can for extra credit (the examples shown above do). You do need to handle multiple operations in the same expression. You will definitely want to keep your math expression code separate from the rest of your code so you can test it and get it working without having to run your whole spreadsheet program over and over. You can achieve this by making a separate class to evaluate expressions or by being careful about how you organize code in your formula cell class.

Checkpoint 3 is worth a total of 35 points. You will be able to find a more detailed rubric online but this is the approximate breakdown of points.

6	All functionality from Checkpoints 1 and 2 must still work
4	Can clear the whole sheet and individual cells
4	Can create new formula cells
2	Can display formula cells individually to see the formula
10	Can see evaluated formulas when formula cells are printed in the grid
2	Cell class hierarchy is expanded to include formula cells
3	All code, especially new formula evaluation code, is well-organized into classes and methods
4	Code is readable (indentation, comments, identifier names, etc.)

Checkpoint 4

In Checkpoint 4, you'll expand the capabilities of your formula cells. They'll need to be able to support references to other cells as well as two special keywords, sum and avg.

The sum and avg keywords will calculate the sum or average of a range of cells. You'll specify a range of cells using two cell references with a colon between them. The range is rectangular, so it may be multiple rows, multiple columns, or both, and it will always show up one space after the sum or avg keyword.

Sum and average commands are part of formula cells. They can be the only thing in a cell, or they can be surrounded by other operations. When there are other operations present, sum and avg should be evaluated first.

Here are some example usages of the sum and average commands (assume some commands were already issued to populate the other cell values you see):

```
Enter a command: C7 = (sum B3:C5)
Enter a command: C8 = (avg B3:C5)
Enter a command: C9 = (3 * avg B3:C5 + 12)
Enter a command: print
+------------+------------+------------+------------+------------+------------+------------+------------+
|            |     A      |     B      |     C      |     D      |     E      |     F      |     G      |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     1      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     2      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     3      |            |    10.0    |    20.0    |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     4      |            |    15.0    |    19.0    |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     5      |            |    22.0    |    8.0     |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     6      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     7      |            |    sum:    |    94.0    |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     8      |            |    avg:    |15.66666666>|            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     9      |            |   more:    |    59.0    |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     10     |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
```

To implement sum and average, you'll need to add some code to your formula cell class. You will probably need to pass your Spreadsheet object into its constructor so the formula cell can use the spreadsheet to see what cells are in different ranges, which may require a bit of refactoring of your TextExcel, Spreadsheet, and/or CellFactory classes.

Sum and average should work on any range of cells that contains only other numeric cells, like number or formula cells. A formula that refers to a range of cells that includes empty, string, or date cells is not valid input. One interesting way to do this is to use an interface to label some types of cells as numeric and use the instanceof operator to see if all the cells in the requested range are numeric. Doing this isn't required but will help you understand what interfaces are good for.

A range can be horizontal, like B3:C3, vertical, like B3:B5, or rectangular, like B3:C5. When you see formulas that include sum or average with other operations, evaluate the sum or average first.

In addition to sum and average, your formulas need to support references to other numeric cells (number and formula cells). Cell references can show up anywhere in your formulas that a number would normally show up. When the spreadsheet is printed, you'll need to replace the references in your formulas with the values of the corresponding cells before evaluating and displaying the formulas. Note that this means you must re-evaluate the formula each time it is printed—you can't just evaluate it once when the cell is constructed, because the cells it refers to may change later.

```
Enter a command: F4 = 12
Enter a command: F5 = 2
Enter a command: F6 = (F4 / F5 + F4 * F5)
Enter a command: F6
F6 = (F4 / F5 + F4 * F5)
Enter a command: print
+------------+------------+------------+------------+------------+------------+------------+------------+
|            |     A      |     B      |     C      |     D      |     E      |     F      |     G      |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     1      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     2      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     3      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     4      |            |            |            |            |            |    12.0    |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     5      |            |            |            |            |            |    2.0     |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     6      |            |            |            |            |            |    30.0    |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     7      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     8      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     9      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     10     |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
Enter a command: F4 = 6
Enter a command: print
+------------+------------+------------+------------+------------+------------+------------+------------+
|            |     A      |     B      |     C      |     D      |     E      |     F      |     G      |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     1      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     2      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     3      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     4      |            |            |            |            |            |    6.0     |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     5      |            |            |            |            |            |    2.0     |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     6      |            |            |            |            |            |    15.0    |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     7      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     8      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     9      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     10     |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
```

Checkpoint 4 is worth a total of 35 points. You can find the detailed rubric online but the approximate breakdown will be:

10	Functionality from checkpoints 1-3 still works
8	Formulas that include sum and avg are evaluated and displayed correctly
10	Formulas that include references to other formulas are evaluated and displayed correctly
4	Code is organized into classes and methods that each have a single responsibility
3	Code is readable (comments, indentation, identifier names, etc.)

Final Submission

For your fifth and final submission, the only new functionality you need to add is the ability to sort ranges of cells. ‘sorta’ and ‘sortd’ are new commands the user can specify. They’ll always be followed by a range of cells, like A3:D7. ‘sorta’ should sort the designated range of cells in ascending order, smallest to largest, and ‘sortd’ should sort them in descending order. If the range is rectangular (not a single row or single column of cells), sort in row-major order, which means the ordering should be like you read a book—left to right, then top to bottom.

```
Enter a command: print
+------------+------------+------------+------------+------------+------------+------------+------------+
|            |     A      |     B      |     C      |     D      |     E      |     F      |     G      |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     1      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     2      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     3      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     4      |            |    5.0     |            |    -6.0    |            |    13.0    |    5.0     |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     5      |            |    19.0    |            |    14.0    |            |    22.0    |    61.0    |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     6      |            |    12.0    |            |    0.0     |            |    17.0    |    19.0    |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     7      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     8      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     9      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     10     |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
Enter a command: sorta B4:B6
Enter a command: sortd D4:D6
Enter a command: sorta F4:G6
Enter a command: print
+------------+------------+------------+------------+------------+------------+------------+------------+
|            |     A      |     B      |     C      |     D      |     E      |     F      |     G      |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     1      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     2      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     3      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     4      |            |    5.0     |            |    14.0    |            |    5.0     |    13.0    |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     5      |            |    12.0    |            |    0.0     |            |    17.0    |    19.0    |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     6      |            |    19.0    |            |    -6.0    |            |    22.0    |    61.0    |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     7      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     8      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     9      |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
|     10     |            |            |            |            |            |            |            |
+------------+------------+------------+------------+------------+------------+------------+------------+
```

You've learned several sorting algorithms in class, and you are free to use any of them for this assignment. You cannot use an existing sort method from Java or any library (e.g. Arrays.sort).

Hint: It is possible to sort cells in place in your grid, but it will be much simpler if you first copy the range of cells you need to sort into a 1-dimensional array, then sort the array, then put the cells back into the same range in the spreadsheet.

Part of organizing your code well for this assignment is reducing the amount of duplicated code you have, particularly between sorting ascending and descending sorts. You should be able to use the same code for both with a little bit of extra logic to decide which order you're sorting in based on what the user typed.

You only need to sort numeric cells (formulas and numbers). Input ranges that include empty or non-numeric cells are not valid, so you only need to handle those cases if you want to earn extra credit.

The final submission is worth a total of 30 points. You can find the detailed rubric on moodle but the approximate distribution will be:

10	All functionality from Checkpoints 1-4 still works
10	Can sort ranges in ascending or descending order
5	Code is well organized into classes and methods
5	Code is readable and clear

Extra Credit Opportunities

You can earn extra credit on this assignment in a variety of ways.

- Sorting non-numeric and/or empty cells
- Implementing a help command that prints out help information for the user. A typical way to organize this is to have 'help' print out the summary of available commands, and commands like 'help print' and 'help clear' and print out more details and examples about those commands.
- Handling formulas with circular references without crashing, e.g. A1 = (A2), A2 = (A1).
- Handling correct order of operations (multiplication and division before addition and subtraction) in formulas.
- Handling errors, so that bad input like "hi program" causes an error message rather than a crash.
- Doing anything else you think of… talk to a teacher if you have an idea, or just do it!

To get extra credit, you will need to submit your code again to the TextExcel – Extra Credit assignment. Be sure your TextExcel class includes a comment at the top listing the things you think should earn you extra credit. If you do not explain your extra credit features in a comment, you will not get any extra credit for them.