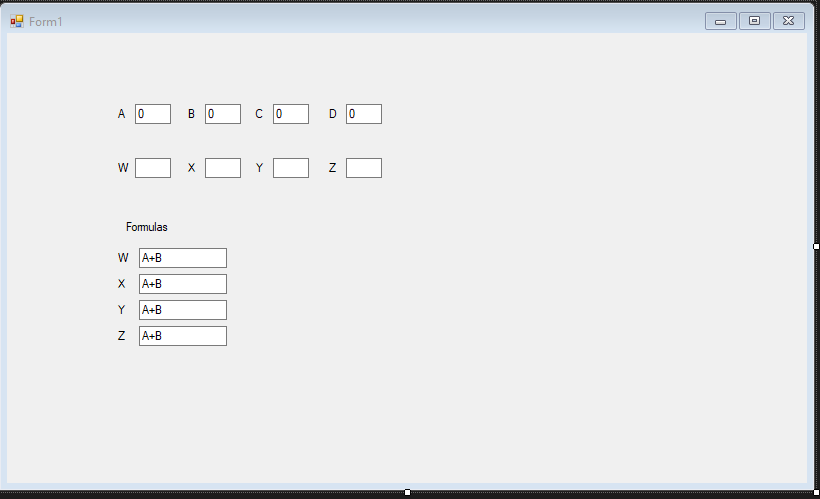
# Lab Exercise #2: Mini Excel

The goal of this lab is to be able to take some sample functions which provide part of the functionality you need, and plan an event-driven program that makes use of those functions (and likely some of your own) to solve a problem.

## Core Task

Create a functioning mini-excel program. Don’t worry, I’ve built / written a lot of it for you to help get you started. The basic interface is below.



When any of the values in A, B, C, or D change, then the formulas in the ‘Formulas’ section should be used to fill out boxes W, X, Y, and Z.

For example, if I edit the boxes so that A = 1, B = 2, C = 4 and D = 8, then change the formulas so that

W=A+B, X=B+C, Y=C+D, and Z = A+D, then the result boxes W, X, Y, and Z should show:

W=3, X=6, Y=12 and Z=9.

Wait a minute, this sounds \*super\* hard. Yes, from scratch it is, but I’ve built the above form and a couple of functions to get you started.

Also, the only formulas that you have to be able to handle for the core task are 3 characters long: a letter A, B, C, or D, the ‘+’ sign, and then a second letter, A, B, C, or D. That is, each formula is only a single addition of two of the input boxes.

## Existing Code

In Module 2, you should see a link for “Code: Lab 2 Exercise Starter Package: MiniExcelStarterCode”.

If you read through the Lab 2 Notes (or remember what we looked at in class), you should be able to download the MiniExcelStarterCode solution and open it in Visual Studio.

## Pre-built Interface

To start with, the interface is built, which is great.

If you open up the Form1.Designer.cs and expand the Windows Form Designer generated code, you will see that the top row of textboxes are helpfully named textBoxA, textBoxB, textBoxC, and textBoxD.

The row of result textboxes are helpfully named textBoxW, textBoxX, textBoxY, and textBoxZ.

The formula textboxes are helpfully named textBoxFormulaX, textBoxFormulaY, textBoxFormulaZ, and, less helpfully, TextBoxFormulaW, with a capital T for no reason. Sorry about that last one.

## Helpful Functions

If you open up Form1.cs, you’ll see that while there aren’t any event handlers built, you have a couple handy functions to start with.

### private double getValue(char boxName)

If you give this function a letter, ‘A’, ‘B’, ‘C’, or ‘D’, it will look at the right textbox and give you back the value in that textbox.

That is, int valueOfBoxC = getValue(‘C’); will get you the value from textBoxC, return it from the function, and the code will put that value in the variable ‘valueOfBoxC’

parseFormula(TextBox formulaBox, int n)

If you give this a TextBox and the number 1, you will get back the first character of the formula in that textbox. If you give this a TextBox and the number 2, you will get back the third character of the formula in that textbox.

There are also two helper functions, getFirstFormulaBoxName and getSecondFormulaBoxName; they will go find the firstName and the secondName from a formulaBox for you.

## How Do I Use Those Functions

I encourage you to look at the code to see how they work.

However, super hint below:

I know that textBoxFormulaX is the name of the TextBox that holds the formula for Box X. If I want to set the value of box X, what do I need? First, I need to know the names of the two textboxes to add together. If textBoxFormulaX has ‘C+D’ in it, then getFirstFormulaBoxName(textBoxFormulaX) will get me ‘C’, and getSecondFormulaBoxName(textBoxFormulaX) will get me ‘D’.

So by calling those functions I know that I need to add together ‘C’ and ‘D’. But those are just the characters that represent those textboxes. And calling the function getValue(‘C’) will give me back the value that is in textBoxC – if I do the same for ‘D’, then I have the two numbers to add together.

And we \*know\* that the result of textBoxFormulaX should go in textboxX

## Getting Started

If you double-click on textBoxD, it will create a textBoxD\_TextChanged event. Any time A, B, C, or D change, you need to recalculate (for now, don’t recalculate when changing a formula, just when you change a value).

You probably \*don’t\* want to write out how to recalculate the spreadsheet 4 times, so you probably \*do\* want to create a recalculate() function and call inside each of the four event handlers.

Inside that recalculate() function, you need to calculate the values for W, X, Y, and Z.

Remember, though, if you know the name of the formulaBox and the outputTextbox (which you do in each of the four cases), you can ‘getValue’ of the ‘getFirstFormulaBoxName’, and ‘getValue’ of the ‘getSecondFormulaBoxName’, then add them together and set the Text of the outputTextBox to the answer.

Completing the core task correctly is worth 1.5/2 for the Lab Exercise

## Extensions

Here are some fun extensions to the core task. You must complete and submit at least one of them to get full marks on the Lab Exercise. I encourage you to either try or at least think about more than one of them.

* Create a double calculateFormula(TextBox formulaBox) function.
  + While solving the core task, you almost certainly wrote some redundant code.
  + You can reduce that immensely by adding a helper function like the above that given a formula textbox object, calculates the value of the formula in that textbox.
* Add a fifth input box
  + E.g. add a label and textBoxE and make sure that formulas like B+E work
* Make the formula work for \* as well as ‘+’
  + Hint: There is already code in parseFormula that pulls the ‘op’ out of the string. You’ll have to change the function logic slightly to give you a way to get the ‘op’ back from that function (perhaps when n== 3??). And of course, do different math depending on the op.
* Make the formula work for a 5-character string
  + E.g. A+B+B is then a valid formula
  + Hint: you’ll need to extend the parseFormula function to let you ask for the 5th character of the string
* Instead of having 4 event handlers calling recalculate, make your recalculate function the only event handler

## Submissions

Please submit the Form1.cs file and the Form1.Designer.cs file (although in this case only two of the extensions require you to change Form1.Designer.cs, but in general that will be good practice).