# Loading and Saving Spreadsheet Data
# Cpt S 321 Homework Assignment
# Washington State University

## Submission Instructions: (see syllabus)

## Assignment Instructions:

**Read each step's instructions *carefully* before you write any code.**

In this assignment, you will add loading and saving capabilities to your spreadsheet application. If you created a Workbook class for the previous assignment (or would like to create one for this assignment) then add Load and Save methods to it. Otherwise you can add these methods to your Spreadsheet class.

Design an XML format for your spreadsheet data. At a high level, it will probably have a structure somewhat like:

```
<spreadsheet>
 <cell name="B1">
  <bgcolor>FF8000</bgcolor>
  <text>=A1+6</text>
 </cell>
</spreadsheet>
```

You'll obviously have more than one cell in most cases. Make sure you do the following:

- Provide saving and loading functions that take a stream as the parameter.
- Add menu options in the UI for saving and loading.
- Make sure the saving and loading code is in the logic engine.
- Use existing XML classes from the .NET framework.
- When saving, only write data from cells that have one or more non-default properties. This means that if a cell hasn't been changed in any way then you don't need to write data for it to the file.
- Clear all spreadsheet data before loading file data. The load-from-file action is NOT a merge with existing content.
- Clear the undo/redo stacks after loading a file.
- Make sure formulas are properly evaluated after loading.
- You may assume only valid XML files will be loaded, but make sure loading is resilient to XML that has different ordering from what your saving code produces as well as extra tags. As a simple example, if you're always writing the <bgcolor> tag first for each cell followed by the <text> tag, then your loader must still support files that have these two in the opposite order. Also, if you didn't write more than these two tags within the <cell> content, your loader should just ignore extra tags when loading. See the example below.

- Use XML reading/writing capabilities from .NET. Do not write your own XML parsers that do things manually down at the string level. We discussed several options in class, such as XDocument, XmlDocument, XmlReader, and XmlWriter.

If you saved:

```
<spreadsheet>
 <cell name="B1">
  <bgcolor>FF8000</bgcolor>
  <text>=A1+6</text>
 </cell>
</spreadsheet>
```

then you must be able to load:

```
<spreadsheet>
 <cell unusedattr="abc" name="B1"> <text>=A1+6</text>
  <some_tag_you_didnt_write>blah</some_tag_you_didnt_write>
  <bgcolor>FF8000</bgcolor> <another_unused_tag>data</another_unused_tag>
 </cell>
</spreadsheet>
```