You may notice that the O (output) in High Level File I/O is left out of this homework. This is because `xlswrite()` does not work on Macs. For your convenience, we do not require you to use this function, except on the ABCs. However, you will still need to know the semantics of this function for the tests. We have set up some of the functions such that the output is a cell array that could readily be written to an Excel file, we just don't require that you actually write any files. If you need to check your code for the ABCs and you have a Mac, you can run the code on a library computer or use Virtual Lab. You will still be responsible for knowing how to use `xlswrite()` on exams.

Happy coding!
~Homework Team

Function Name: `boneless`

Inputs:

1. (*cell*) Cell array of the toppings on your pizza

Outputs:

1. *(cell)* Updated cell array of toppings on your pizza
2. *(char)* A string stating how many bones you found in your pizza

Background:

It's 10 p.m. and you're starving, but you're tired of dining hall food. You call your local pizza place and order a 🅱oneless pizza with a 2-liter Coke, but the employee at the pizza place tells you that the 2-liter machine broke, and that they only have 1-liter Coke. Suffering from a severe case of test week, you take all your anger and frustration out on the poor employee, and as revenge, he delivers your pizza 🅱one-in. Luckily, though, you can use your MATLAB skills to replace the bones in your pizza with the pineapple you have in the fridge!

Function Description:

Given a cell array of all the toppings on your pizza, replace all occurrences of the character vector `'bones'`, case insensitive, with `'pineapple'` **without** using iteration. Output the updated array, and a string of the format `'What a shame, I found <num> bones in my pizza.'` where `<num>` is the number of cells that contained bones. If there is only one cell containing the string `'bones'`, your output string should be `'What a shame, I found a bone in my pizza.'`. If your pizza was delivered 🅱oneless, output the string `'I found no bones in my pizza!'`.

Example:

```
>> pizza = {'PePperonI';'BONES';tomato sauce'}
>> [newPizza, str] = boneless(pizza)
      newPizza → {'PePperonI';'pineapple';'tomato sauce'}
      str → 'What a shame, I found a bone in my pizza.'
```

Notes:

- Your function should not be case sensitive. For example, the string `'BoNeS'` should still be replaced with `'pineapple'`.
- Each cell is guaranteed to have only one topping.
- Apart from the replaced cells, your output cell array should be identical to the input cell array.

Hints:

- Think about how you can use `strcmpi()` to solve this problem.

Function Name: `cowcium`

Inputs:

1. (*char*) The name of an Excel file containing cow data
2. (*char*) The column header to search for

Outputs:

1. (*cell*) The updated cell array

Background:

You are a worker in one of the most noble professions a person can undertake: calcium farming. Er, dairy farming. In order to deliver calcium and strong bones to people all across the country, you have to keep close watch over your cows, collecting data to track the cows over time and making sure that your cows produce enough calcium for public consumption.

Function Description:

Read in the Excel file given by the first input. The Excel file contains headers in the first row, and data for each cow in the subsequent rows. First, sort the rows of the array in descending order based on the data in the column identified by the second input. Second, remove the rows corresponding to any cows that have a score of less than 20 in the `'Calcium'` column, and output this updated cell array.

Example:

`'Cows.xlsx'` →

'Calcium'	'Cuteness'	'Name'
15	10	'Petunia'
30	11	'Rose'
45	9	'Lily'

```
>> [out] = cowcium('cows.xlsx', 'Cuteness')
      out →
            [ {'Calcium'}    {'Cuteness'}     {'Name'};
                 {30}           {11}          {'Rose'};
                 {45}            {9}          {'Lily'}]
```

Notes:

- The data in the column specified by the second input is guaranteed to be numeric.
- `'Calcium'` and the header specified by the second input are guaranteed to appear in the first row of the Excel file.

Function Name: `shinyTeeth`

Inputs:

1. (*cell*) A 1XN cell array representing someone's teeth.

Outputs:

1. *(double*) The positions of the cavities in the teeth.
2. (*double*) The total strength of the teeth.

Background:

Before every concert at the Dimmsdale Dimmadome owned by Doug Dimmadome, Chip Skylark makes sure to get a checkup at the Dimmsdale Dentist. He has to make sure his teeth are all nice and strong with no cavities before he sings his hit bop "My Shiny Teeth and Me." You'll use MATLAB to perform a dental exam on his teeth!

Function Description:

A set of teeth is represented by a 1XN cell array. Each cell in this cell array is a nested cell representing a tooth. The cells are nested a varying amount of times. After you peel through all the layers of a tooth there will be a non-negative value representing that tooth's strength. If a tooth has a 0, then that tooth has a cavity. Return the positions of the cavities and the total strength of the teeth.

Example:

```
      teeth = [{{{{4}}}}, {{{1}}}, {{{0}}}]
[cavities, strength] = shinyTeeth(teeth)
>> cavities = 3;
>> strength = 5;
```

Notes:

- If there are no cavities, return an empty vector.
- Inside of each entry in the 1xN cell array, there will be no nested arrays. This means that, underneath however many layers, the eventual contents will be 1x1.

Function Name: `malk`

Inputs:

1. (*char*) The filename of a text file containing your shopping list
2. (*char*) The filename of an Excel file containing a grocery store inventory

Outputs:

1. *(char*) A string stating the total cost of your grocery store run

Background:

It's Sunday night and you need a late night snack before you finish your CS 1371 homework. You pour a bowl of some Lucky Charms and just need to grab that sweet CAlcium - but wait!! You milk has expired! It has gone bad - it has turned into: MALK! Now you need to go to the grocery store and buy more milk, among other things.

Function Description:

Given a grocery list text file with each line in the form: `'<item> <number of items>'` and an excel file that contains an inventory for the grocery store you are visiting. The first column has the number of items in stock, the second column will have the item name, and the third column has the price per unit (in dollars). Your job is to calculate the total cost of your grocery store run and output the string `'My total will be $<total>.'`, where total always has 2 decimal places.

Example:

If the store's inventory is

8	Apples	2.08
5	Milk	4.02
36	Eggs	.72

and your grocery list has,

Milk 1

eggs 12

Then your function should output:

```
'My total will be $12.66.
```

Notes:

- The grocery list will have only 1 item per line, and each item will be one word
- Case does not matter
- You are guaranteed to find everything you need and the store will always have enough
- You do **NOT** have to update the store's inventory

Hints:

- You should use %0.2f to put a number with 2 decimal points in a string.
- Remember how certain string functions work on cell arrays.

Function Name: `calcium`

Inputs:
1. (*char*) The name of a text file to read

Outputs:
1. *(cell)* A 2xN cell array describing the atoms appearing in the file

Background:

Calcium is used in making porcelain and glass, in purifying sugar, in medicines, and in much more. You work in procurement at a chemical manufacturing plant. The manufacturing technicians send you a weekly list of all the compounds they need to make, and you use this information to decide how much of each pure element to buy that week. Obviously, you use MATLAB to complete such a task.

Function Description:

Write a function that reads a text file full of chemical formulas and creates a cell array from the information in the text file. The first row of the cell array should contain all the atoms that appear in the text file (one atom per cell), and the second row should contain the number of times that atom appeared. Your output should contain all the atoms sorted in alphabetical order.

Example:

```
formulas.txt →          atoms = calcium('formulas.txt')
CaCO3                   atoms => {'C'    'Ca'    'Cl'    'H'    'O'
CaCl2                              13      5       2      24     20}
CaOHOH
CaO
C12H22CaO14
```

Notes:
- Atoms will always be represented as their chemical symbols, which will always contain either one or two letters. All chemical symbols begin with a capital letter and end with a lowercase letter if a second letter is present.
- Each line in the text file will contain only letters and numbers with no spaces or special characters.

Hints:
- You may find the `strcmp()` function useful in determining whether a particular atom should be added to the cell array.
- If you encounter an atom that is already in the cell array, simply update the count for that atom.
- Consider making a helper function to update your cell array.

- You may find the find() function useful