Notice

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.

Happy Coding,
~Homework Team

Function Name: `sellery`

Inputs:

1. (*char*) The name of an Excel file containing stand data

Outputs:

1. *(char*) A string with the total profits

Background:

You have made it halfway through the semester and realize that it's time to start making a little extra money. To do this, you and your friends decide to set up stands around campus selling celery sticks. Due to your questionable arithmetic skills, you decide to use your MATLAB skills to keep track of your monthly profits.

Function Description:

Given an Excel file with all the data on your stands' revenue and expenditures for the semester, calculate the overall profits.

If your profit is non-negative, output the string: **`'You made a profit of $<value>!'`**
If your profit is negative, output the string: **`'You are $<value> in debt!'`**

Example:

Stand	Revenue	Expenditure
CULC	13	26
CoC	80	9
CoB	54	44
Student Center	23	12
CRC	43	10

Output => **`'You made a profit of $112.00!'`**

Notes:

- The spreadsheet will always have at least two columns with subject headers **`'Revenue'`** and **`'Expenditure'`**.
- The columns can be in any order.
- You can use **`%.2f`** to display two decimal places in a formatted string.

Function Name: `celery`

Inputs:

1. (*cell*) An Nx3 cell array containing the names, dips, and rankings of the party-goers

Outputs:

1. *(cell*) An Nx2 cell array containing the names of your friends and their preferred dip

Background:

Your stellar MATLAB skills got you a recent raise on your *sellery* so you decide to throw a party where a *bunch* of your friends come together in your *cellar* for a potluck dip fiesta to try new dips and *stalk* your exes. Each of your friends brings a dip and ranks the dips in order of their preference.

Function Description:

Given an n by 3 cell array with your friends' names in the first columns, the dip that they brought in the second, and a vector of their rankings in the third, output a new cell array with the name of your friends in the first column and their top-ranked dip in the second. The vector of rankings will be in the same order as the dips (the first index in the vector will correspond with the first friend/dip and so on).

Example:

`'John'`	`'Peanut Butter'`	`[1 3 2]`
`'Paul'`	`'Ranch'`	`[3 2 1]`
`'Ringo'`	`'Hummus'`	`[2 3 1]`

out⇒

`'John'`	`'Peanut Butter'`
`'Paul'`	`'Hummus'`
`'Ringo'`	`'Hummus'`

Notes:

- There will be no ties; each friend will rank only one dip as first

Function Name: `nationalCelery`

Inputs:

1. *(char)* The name of the file containing which letters to extract
2. *(cell)* A 16x8 cell array containing the Declaration of Celery

Outputs:

1. *(char)* A string containing the decoded message

Background:

Back in 2004, possibly the greatest actor of our time, Nicolas Cage, starred in possibly the greatest movie of our time, *National Treasure*. Three years later, the second greatest movie, *National Treasure: Book of Secrets* starring Nicolas Cage, took the world by storm. In the ensuing ten years, Nicolas Cage has found that his life has become increasingly dull, as he hasn't been able to star in a 3rd, 4th, 5th, or even a 6th *National Treasure* movie. Yearning for adventure, Nicolas looks to the two movies that gave a new meaning to the word "perfection" for guidance. Ben Gates had sought after the greatest national treasures, the Declaration of Independence and the City of Gold. Without a moment of hesitation, Nicolas Cage knew what he needed to feel complete again. Yes, you guessed it; Nicolas Cage is going to steal the celery from the White House vegetable garden.

Function Description:

After extensive research, Nicolas Cage learns that the secrets of the celery are encoded in the Declaration of Celery. This document is stored in a cell array where each cell contains a string that makes up part of the Declaration of Celery. The text file contains the "coordinates" for the cipher. Each line will correspond to a specific letter, and will be formatted as follows:

```
<row number>-<column number>-<character number>
```

For example, the line

```
'5-3-10'
```

Corresponds to the 5th row and 3rd column of the cell array, and 10th character of that string. Additionally, instead of the above format, a line could just be the string

```
'space'
```

which means to insert a space. You are to use the information in the first file to extract a message from the cell array. The output is a string containing the letters and spaces pulled from the second file in the order that they are in the cipher file.

Notes:

- The letter "coordinates" are guaranteed to be valid (i.e. we won't ask for the 5th character of a line that has only 4).

Function Name: **`celeryComp`**

Inputs:

1. (*char*) The name of an excel file containing the different kinds of celery foods
2. (*cell*) a Nx1 cell array containing a ranking of celery based or related foods

Outputs:

1. *(cell*) (2^M) x (M+1) A cell array of the completed bracket

Background:

You and your friends are discussing which of your favorite celery dishes is the absolute best. However, everyone thinks their own amazing celery-ific dish is the most amazing celery-ific dish ever! You, being the MATLAB wiz you are, decide to code a tournament bracket to decide whose dish is the most amazing celery-ific dish present. To prepare you find a Huffington Post article about 'Celery Recipes That Are Freakishly Delicious' as well as other similar rankings and decide to use something similar to judge your grand celery tournament!

Function Description:

Write a function that takes in your friend's celery dishes and rankings of various celery dishes and outputs a cell array of the finished bracket. The number of recipes will always be a power of 2 so that pairs can be formed until there is one winner left. The excel file will be of size 2^M x 1, thus your final bracket will be of size (2^M) x (M+1). When putting the winner of each match in your cell array, justify all winners to the top, as shown below:

Excel file (order tells which dishes compete)

`'H'`
`'G'`
`'F'`
`'E'`
`'D'`
`'C'`
`'B'`
`'A'`

Cell Array (order tells the rank of each dish which determines what will win each round)

`'A'`
`'B'`
`'C'`
`'D'`
`'E'`
`'F'`
`'G'`
`'H'`

Continued...

Output (completed bracket)

'H'	'G'	'E'	'A'
'G'	'E'	'A'	''
'F'	'C'	''	''
'E'	'A'	''	''
'D'	''	''	''
'C'	''	''	''
'B'	''	''	''
'A'	''	''	''

To decide who wins the match, use the given cell array. The first row in the cell array is the 'best' dish, and the last row is the "worst" dish. Every dish is in order between them, and there will be no ties. The length of the cell array may have extra rankings of dishes and be longer than the number of entries in your excel document; however, it will never be shorter.

Notes:

- The pairs of adjacent dishes in each column will be the ones competing with each other. For the first round in the above example, 'H' competes with 'G', 'F' competes with 'E' and so on. In the second round, 'G' competes with 'E', and 'C' competes with 'A'.
- There will always be 2^M teams so the bracket will work out properly.
- If you are unfamiliar with brackets, check out this link.

Hints:

- The total number of columns in your final bracket will be equal to **`log2(<number of rows in original bracket>) + 1`**.

Function Name: `countCelery`

Inputs:

1. *(char)* The filename of the text file that you want to count the words in

Outputs:

1. *(cell)* A 2xN cell array containing every unique word in the first row and the number of times it appears in the text in the second row
2. (*double*) The number of times the word '**`celery`**' appears

Function Description:

Have you ever wondered how many times a word has been said in a book? Or wanted to see if you kept using the same adjectives in your english paper? Or perhaps wanted to know how many times a text file contained the word '**`celery`**'? With the powerhouse that is MATLAB, you now can!

You will be given a **`.txt`** file, and extract every unique word and store them in a cell array, with the first row of the cell array containing the word, and the second row containing the number of times that word appears in the text. In other words, each column should have a word with the corresponding number of appearances in the text. Case should be ignored when looking for unique words, and any characters that are not letters or spaces should be removed. The words should appear in alphabetical order in the output cell array. You should also output the total number of times the word 'celery' appears

Example:

```
Twinkle.txt → 'twinkle 1738 twinKLe## little celery'

            out1 =
                 {'celery', 'little', 'twinkle'
                      1,        1,        2    }
            out2 = 1
```

Notes:

- The words in the output cell array should be lowercase.
- If you want to test different files, or investigate a specific piece of work, check out archive.org and search for your text of choice.

Hints:

- It may be useful to initialize the cell array, among other items, before you enter a loop.
- Remember that many functions that work with strings will also work on cells containing strings. In fact, some functions that don't work on strings will work on a cell array containing strings, like **`sort()`.**
- Cell arrays are just like normal arrays in the sense that you can delete and add/expand