Function Name: `calcGravity`

Inputs:

1. (*double)* A vector of masses of a set of objects
2. *(double)* A vector of masses of another set of objects
3. *(double)* A vector of the distances between the pairs of objects

Outputs:

1. (*double*) A vector of the gravitational force values between the objects

Background:

You are working on your physics homework, but eventually realize you would much rather be doing CS 1371 homework instead. The only obvious solution? Turn your physics homework into CS 1371 homework! Use MATLAB to solve your physics homework problems by calculating the gravitational force between objects.

Function Description:

You will be given 3 input vectors: 2 sets of objects and the distances between them. For example, the first value in each of the first two vectors each correspond to the mass of an object, and the first value in the distance vector corresponds to the distance between those two objects. Use element by element operations to calculate the gravitational force between each of the pairs of objects with the following formula:

$$F_g = \frac{Gm_1m_2}{d^2}$$

Use `6.67e-11` as the gravitational constant G. Round to 4 decimal places.

Example:

```
mass1 = [100000, 200000, 300000];
mass2 = [800000, 900000, 1000000];
distance = [1, 2, 3];
Fg = calcGravity(mass1, mass2, distance);
Fg → [5.3360, 3.0015, 2.2233]
```

Notes:

- You can use scientific notation to type in the gravitational constant, e.g. you can type `'6.67e-11'` as shown, and MATLAB will know what you mean.

Function Name: `formatBookTitle`

Inputs:
1. (*char*) An incorrectly formatted book title

Outputs:
1. (*char*) A correctly formatted book title

Background:

You love fiction books. You also love to recommend your favorite books to your friends. But you also realize that you are too lazy to properly format the title of your favorite books by hand. You decide to create a MATLAB function to do it for you.

Function Description:

To generate the correctly formatted title of a book, follow these rules:

1. Capitalize the first letter of each word unless they are the following words: `'and'`, `'the'`, `'of'`.
2. The first letter of the character vector should always be capitalized, even if it is `'and'`, `'the'`, or `'of'`.

Example:

```
unformatted = 'THe OLd MaN AND tHe sEa';
formatted = fixBookTitle(unformatted);
formatted → 'The Old Man and the Sea'
```

Notes:
- All words will be separated by a space.
- The words `'and'`, `'the'`, and `'of'` will not appear at the beginning of other words.

Hints:
- Remember that the first letter of a word follows a space except when the words is the first word in the string.
- `strfind()` will always return the index of the first letter of the string you are trying to find.
- Think of how you can combine the `upper()` and `lower()` functions with indexing to capitalize or lowercase a singular or set of letters.

Function Name: `repWord`

Inputs:
1. (*char*) A sentence to be modified
2. (*char*) The word to be replaced in the sentence
3. (*char*) The word to replace with

Outputs:
1. (*char*) Modified sentence

Banned Functions:

`strrep()`

Background:

The ctrlF and replace feature on your laptop is broken and you need to use MATLAB to do the job!

Function Description:

Your function will replace one word within your sentence with another, indicated by your input variables. Your output will be the entire modified sentence with the replaced word.

Example:

```
original = 'My favorite class is physics!';
wordFind = 'physics';
wordReplace = 'CS1371';
out = repWord(original, wordFind, wordReplace);
out →  'My favorite class is CS1371!'
```

Notes:
- There will be exactly one instance of the word you will replace (it will not appear twice in a sentence)
- Your function should be case sensitive (i.e. `'MATLAB'` and `'matlab'` are not the same)

Hints:
- The strfind() function will be useful!

Function Name: `wordLen`

Inputs:

1. (*char*) A character vector of three words

Outputs:

1. (*double*) A 1x3 vector with the lengths of each word

Background:

You know how sometimes you just really want to know the length of a word, but for specifically three word sentences? You don't?! Oh well, let's make a function to do it anyway.

Function Description:

Given a sentence of three words separated by spaces, find the length of each word and output the values as a vector.

Example:

```
lens = wordLen('Matlab rocks dude')
lens → [6 5 4]
```

Notes:

- Words will only be separated by spaces.
- There will never be any punctuation.

Function Name: `dataMixer`

Inputs:

1. (*double*) A 1xN vector to be inserted into odd indices
2. (*double*) A 1xN vector to be inserted into even indices

Outputs:

1. (*double*) A 1x2N vector of your compiled data

Background:

You are a brand new data scientist and have just received your first project: you must use MATLAB to create a data set to be presented at next week's conference. The only problem is that the researcher you are partnered with gave you data that's all out of whack!

Function Description:

You are given two vectors of the same length that must be sorted and then combined. The first vector you are given should be sorted in **ascending** order. The second vector should be sorted so that its new order follows that of the first (i.e. if the 4th index in the first vector becomes the 1st index after sorting, the 4th index in the second vector should also become the 1st index after sorting). Next, combine your two sorted vectors into a single vector of length 2N, where the odd indices are filled with values from the first vector and the even indices are filled with values from the second vector.

Example:

```
data = dataMixer([2 7 1 9], [5 6 3 1]);
data → [1 3 2 5 7 6 9 1]
```

Hints:

- What does each output of `sort()` give you?