Assignment 1
Harvesting Emails and Phone Numbers
Using Regular Expressions

Your goal in this assignment is to write regular expressions that extract phone numbers and regular expressions that extract email addresses. To start with a trivial example, given

```
jurafsky@stanford.edu
```

your job is to return

```
jurafsky@stanford.edu
```

But you also need to deal with more complex examples created by people trying to shield their addresses, such as the following types of examples:

```
jurafsky(at)cs.stanford.edu
jurafsky at csli dot stanford dot edu
```

You should even handle examples that look like the following

```
<script type="text/javascript">obfuscate('stanford.edu','jurafsky')</script>
```

For all of the above you should return the corresponding email address

```
jurafsky@stanford.edu OR jurafsky@cs.stanford.edu OR jurafsky@csli.stanford.edu
```

as appropriate.

Similarly, for phone numbers, you need to handle examples like the following:

```
TEL +1-650-723-0293
Phone:  (650) 723-0293
Tel (+1): 650-723-0293
<a href="contact.html">TEL</a> +1 650 723 0293
```

all of which should return the following canonical form:

```
650-723-0293
```

(you can assume all phone numbers are inside North America).

You will be given some data to test your code on together with the correct answers, so that you can make sure your code extracts all and only the right information. You should be creative in looking at the web and thinking of different types of ways of encoding emails and phone numbers, and not just rely on the methods listed in this document.

When compiling your program, the code will compare your results with the ones in the devGOLD file. The true positive section displays e-mails and phone numbers which the starter code correctly matches, the false positive section displays e-mails which the starter code regular expressions match but which are not correct, and the false negative section displays e-mails and phone numbers which the starter code did not match, but which do exist in the files. Your goal, then, is to reduce the number of false positives and negatives to zero.

The function

```
def processFile(name,f)
```

has been flagged for you with a "TODO" marker. This function takes in a file name and a file object and returns a list of tuples representing e-mails or phone numbers found in that file. Specifically the tuple has the format:

```
(filename, type, value)
```

where type is either 'e', for e-mail, or 'p' for phone number, and value is just the actual e-mail or phone number. Your job is to build a system of regular expressions which output as many correct contact tuples as possible.

The canonical forms we expect are:

```
user@example.com
650-555-1234
```

The case of the e-mails you find should not matter because the starter code and the grading code will lowercase your matched e-mails before comparing them against the gold set.