CSCE-638, Programming Assignment #1 SpamLord
Due: Wednesday, Sep. 11, 2018 by 11:00pm

Here's your chance to be a SpamLord! Yes, you too can build regexes to help spread evil throughout the galaxy!

More specifically, 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:

```
huangrh@cse.tamu.edu
```

you should return:

```
huangrh@cse.tamu.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 that I'm sure you've all seen:

```
huangrh(at)cse.tamu.edu
huangrh at cse dot tamu dot edu
```

You should even handle examples that look like the following (as it appears on your screen; we've used metachars on this page to make it display properly):

```
<script type="text/javascript">obfuscate('cse.tamu.edu','huangrh')</script>
```

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

```
Phone:  (979) 862-2908
Tel (+1):  979-862-2908
<a href="contact.html">TEL</a> +1&thinsp;979&thinsp;862&thinsp;2908
```

all of which should return the following canonical form:

```
979-862-2908
```

(you can assume all phone numbers we give you will be inside North America). In order to make it easier for you to do this, we will be giving you some data to test your code on, what is technically called a development test set. This is a document with some emails and phone numbers, together with the correct answers, so that you can make sure your code extracts all and only the right information.

You will be graded on how well your regular expressions find emails and phone numbers in a different test set that we have. Because you don't know exactly what trickery goes on in this test set, 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 we've listed here.

Note you won't have to deal with: really difficult examples like images of any kind, or examples that require parsing names into first/last like:

```
"first name"@cse.tamu.edu
```

or difficult examples like

```
To send me email, try the simplest address that makes sense.
```

1. Python Starter Code

You are encouraged to use Python for the programming assignments in this class, but if you prefer, you can use Java too. However, simple starter code will be provided in some of the programming assignments (including this one) and only in Python.

In the same directory where the provided code and data are, you can run the code:

```
python SpamLord.py data_dev/dev/ data_dev/devGOLD
```

The function

```
def process_file(name, f):
```

has been flagged for you with the universal "TODO" marker. This function takes in a file object (or any iterable of strings) 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.

OUTPUT FORMATTING

The results will look something like the figure in the next page. Even if you program using a language different from Python and do not use the starter code, your program output should follow the same structure.

The true positive section displays e-mails and phone numbers which the code correctly matches, the false positive section displays e-mails which the coded regular expressions match but which are not correct, and the false negative section displays e-mails and phone numbers which the code did not match, but which do exist in the html files. Your goal, then, is to reduce the number of false positives and negatives to zero.

GRADING CRITERIA

Your program will be graded based on both given dev cases (50%, 5 points) and new test cases (50%, 5 points)! **So please test your program thoroughly to evaluate the generality and correctness of your code!** Even if your program works perfectly on the examples that we give you, that does not guarantee that it will work perfectly on additional test cases.
Specifically, the scoring metric will largely depend on the total number of errors (false negatives and false positives) E and also the total number of correct extractions (true positives) C.

```
Dev:
If E < 10:  score(E) = 5 - .3 * E
else if C >= 30 :  score(E) = 2



Test:
If E <= 10:  score(E) = 5
else if 10 < E < 20:  score(E) = 5 - .3 * (E-10)
else if C >= 30:  score(E) = 2
```

ELECTRONIC SUBMISSION INSTRUCTIONS
(a.k.a. "What to turn in and how to tun in")

You need to submit 2 things:

1. The source code files for your program. Be sure to include all files that we will need to compile and run your program!
2. A report file that includes the following information:
   - how to compile and run your code
   - results and analysis
   - any known bugs, problems, or limitations of your program

REMINDER: your program **must** compile and run. We will not grade programs that cannot run.

How to turn in: please submit your program and report in one zip file through ecampus.

```
True Positives (4)
balaji e balaji@stanford.edu
nass e nass@stanford.edu
shoham e shoham@stanford.edu
thm e pkrokel@stanford.edu
False Positives (1)
psyoung e young@stanford.edu
False Negatives (54)
ashishg e ashishg@stanford.edu
ashishg e rozm@stanford.edu
ashishg p 650-723-1614
...
```

Figure 1: Sample Output for SpamLord