Mike Roylance
roylance@washington.edu

Q2: Explanation of algorithm:

Tokenization Criteria
First, I compiled a list of known regexes for the following items: *emails*, *urls*, *American currency*, *hyphenated words*, and *phone numbers*. I then created a couple more regexes, one that matched common words ([A-Za-z0-9]+) and single characters that matched anything but alphanumeric.

Next, I grabbed common abbreviations as found at this site: http://www.indiana.edu/~letrs/help-services/QuickGuides/oed-abbr.html . I wrote a JavaScript file that converted all the abbreviates into a hashtable.

Finally, I wrote an algorithm (in tokenizer.js) that systematically goes through the largest subset of the string and checks the abbreviations and regex matches. For example, let's say I had the string "hello world".

This is how the process would check:
"hello world" -> match any abbreviations or regexs ? add token to list : continue on
"hello worl" -> match ...?
"hello wor" -> match ...?
"hello wo" -> match …?
"hello w" -> match ..?
"hello " -> match .. ?
"hello" -> match ..? In this case, it does match. The algorithm then adds "hello" as a token and starts the process again from the zero-based index of 5.

" world" ? .. match
" worl" … match? etc

Remaining Problems
My tokenizer doesn't handle spaces well. For instance, "New York" will be viewed as two separate tokens. Also, my regex for phone numbers could be more sophisticated.

Resources
My tokenizer uses the following resources:
abbreviations from here (I data wrangled them into a JavaScript hash table):
http://www.indiana.edu/~letrs/help-services/QuickGuides/oed-abbr.html
email regex was one I developed a while ago
url regex from here:
http://net.tutsplus.com/tutorials/other/8-regular-expressions-you-should-know/

currency regex from here:
http://stackoverflow.com/questions/354044/what-is-the-best-u-s-currency-regex
phone regex from here:
http://stackoverflow.com/questions/123559/a-comprehensive-regex-for-phone-number-validation

Q4: Run the code in Q1 and Q3:

- the numbers of tokens in ex1 and ex1.tok

FileName	Number of Tokens
ex1	40362
ex1.tok	41706

- the sizes of ex1.voc and ex1.tok.voc

FileName	Sizes (number of vocabulary words)
ex1.voc	39824
ex1.tok.voc	47493

Q5:

- **The probability mass function for binomial is:**
  - $f(k;n,p) = \Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$
  - So if we spin the wheel 500 times, n = 500. 13 is the exact number we're looking for and p = (1/38)
  - $\binom{500}{13} (\frac{1}{38})^{13} (1 - \frac{1}{38})^{487}$
  - 0.1113
- The poisson distribution function is this:
  - $\frac{\lambda^x e^{-x}}{x!}$
  - lambda = # of occurrences in a given interval
  - we know that the probability of getting a 7 is (1/38)
    - therefore lambda = 500/38 = 13.1579 for 500

changes

- we now calculate P(x=0) + P(x=1)...P(x=13)

P(0)	$\frac{13.1579^0 e^{-13.1579}}{0!}$	0.00000193017
P(1)	$\frac{13.1579^1 e^{-13.1579}}{1!}$	0.00002539704
P(2)	$\frac{13.1579^2 e^{-13.1579}}{2!}$	0.0001670859
P(3)	$\frac{13.1579^3 e^{-13.1579}}{3!}$	0.00073283319
P(4)	$\frac{13.1579^4 e^{-13.1579}}{4!}$	0.00241063645
P(5)	$\frac{13.1579^5 e^{-13.1579}}{5!}$	0.00634378269
P(6)	$\frac{13.1579^6 e^{-13.1579}}{6!}$	0.01391180971
P(7)	$\frac{13.1579^7 e^{-13.1579}}{7!}$	0.02615002871
P(8)	$\frac{13.1579^8 e^{-13.1579}}{8!}$	0.04300993286
P(9)	$\frac{13.1579^9 e^{-13.1579}}{9!}$	0.06288004395
P(10)	$\frac{13.1579^{10} e^{-13.1579}}{10!}$	0.08273693303
P(11)	$\frac{13.1579^{11} e^{-13.1579}}{11!}$	0.09896766283
P(12)	$\frac{13.1579^{12} e^{-13.1579}}{12!}$	0.10851721757

P(13)	$\frac{13.1579^{13}e^{-13.1579}}{13!}$	0.10983528438
		0.55569057848 or 56%

I didn't really believe that answer at first, so I ran it on Octave on Patas and got the same answer:

```
octave:15> for i=0:13
> sum = sum + (lambda^i) *(e^(-1*lambda)) / (factorial(i))
> end
sum =  1.9302e-06
sum =  2.7327e-05
sum =  1.9441e-04
sum =  9.2725e-04
sum =  0.0033379
sum =  0.0096817
sum =  0.023594
sum =  0.049744
sum =  0.092754
sum =  0.15563
sum =  0.23837
sum =  0.33734
sum =  0.44586
sum =  0.55569
octave:16>
```