Assignment 4 (Due: Fri Oct 1 at 5pm) Computational Linguistics (LING 455)

(1) Define a small corpus of **at least 3** sentences, using the newline character `\n` to separate sentences. (Don't put any actual linebreaks in your corpus, or `ghci` will complain.). Try to repeat some common words in different contexts. Feel free to include capitalization and punctuation — you'll be `clean`-ing it later.

```
myCorpus :: String
myCorpus = "REPLACE ME"
```

(2) Now, segment your corpus into its sentences (see `W4.hs` for a function that breaks a `String` on the `\n`'s).

```
mySentences :: [String]
mySentences = undefined
```

(3) Next, `clean` and `tokenize'` each sentence using `map`. Each individual sentence will be transformed into a list of tokens. (Again, see `W4.hs` for help. We are using `tokenize'` so that `<s>` and `</s>` are added as start and stop symbols to each sentence.)

```
myTokened :: [[String]]
myTokened = undefined
```

(4) Next, use `map` to turn each sentence into a list of its bigrams. You can use any of the 3 *n*-gram strategies we discussed. (Keep the bigrams separated by sentence for now!)

```
-- myGrammed :: ?
myGrammed = undefined
```

The type `myGrammed` has will depend on which strategy you chose. Once you've defined `myGrammed`, save and `:r`, ask `ghci` to tell you `:t myGrammed`, and fill it in above. Does the result make sense to you?

(5) `myGrammed` should have separate groups of bigrams, one per sentence. Flatten this out using `concat`, yielding a flat list of all the bigrams in your corpus.

```
-- allBigrams :: ?
allBigrams = undefined
```

Again, the type of `allBigrams` depends on the strategy for extracting *n*-grams you chose. And again, save and `:r`, ask `ghci` to tell you `:t allBigrams`, and fill it in above. Does the result make sense to you?

(6) **Extra credit.** Can the bigrams output of `allBigrams` be used to generate any *new* sentences — that is, any sentences not already in your starting corpus (remember that a path through the bigrams only counts as a sentence if that path begins with `<s>` and ends with `</s>`)? Report one of those sentences if so:

```
novelSentence :: String
novelSentence = undefined
```

If you can't generate any new sentences with the bigrams in `allBigrams`, tweak `myCorpus` until you can. The following code will check that the sentence you report can in fact be generated from your corpus's bigrams (and that it is not already in your corpus):

```haskell
doesItWork :: String -> Bool
doesItWork s = let cleaned  = clean s
                   cleaneds = map clean mySentences
                   sBigrams = bigrams (tokenize' cleaned)
                   oBigrams = concatMap bigrams (map tokenize' cleaneds)
                   tf       = all (\bg -> elem bg oBigrams) sBigrams in
  not (elem cleaned cleaneds) && tf
```