1. Write a program that checks if a word that the user enters is a palindrome. Make sure that your program works for words with an odd number of characters as well as for words with an even number.

2. It turns out that in the task of authorship attribution, it really helps to use character n-grams to classify texts. Take the file `vm.pos` and print out all the unique character 4-grams (i.e., 4-gram types), along with their frequencies.
   - Ignore lines starting with `%%`. Ignore POS information.
   - When you hit a sentence boundary (indicated by an empty line), stop the 4-gram calculation and start over.
   - You should use the NLTK `FreqDist` utility to store the 4-grams. If I were you, I would first make sure I could print out all the character 4-grams in the order they appear in the text. Only after that would I worry about storing them.

3. NLTK, ch. 1, #22 (http://www.nltk.org/book/ch01.html):
   Find all the four-letter words in the Chat Corpus (`text5`). With the help of a frequency distribution (`FreqDist`), show these words in decreasing order of frequency.

4. **Bonus:**
   You’re going to have a user enter a sequence of `a`’s and `b`’s one at a time. Anything other than an `a` or a `b` signals that they are done inputting.
   - At the very minimum, they must enter `ab`. If they don’t, tell them “I’m sorry; that is not a valid string.”
   - They can have as many `b`’s as they want, but every time they enter an `a`, it must be followed by a `b`. If they enter anything other than an `a` after an `a`, stop asking for input and tell them “I’m sorry; that is not a valid string.”
   - They must stop at a `b`, i.e., the string cannot end in `a`. If you are familiar with finite-state machines, this is the same as what is in Figure 1.
   - When the user enters a character other than `a` or `b`, evaluate whether the string is valid and either print “Thank you!” or “I’m sorry; that is not a valid string.”

Here are some example sessions:

```
Please enter a letter: a
Please enter a letter: b
Please enter a letter: b
Please enter a letter: END
Thank you!
```
Figure 1: A finite-state machine accepting \((ab^+)^+\)

Please enter a letter: a
Please enter a letter: a
I’m sorry; that is not a valid string.

Please enter a letter: a
Please enter a letter: b
Please enter a letter: b
Please enter a letter: a
Please enter a letter: END
I’m sorry; that is not a valid string.

Please enter a letter: a
Please enter a letter: b
Please enter a letter: a
Please enter a letter: b
Please enter a letter: a
Please enter a letter: b
Please enter a letter: c
Thank you!