1. Logon to ieng6 and execute the following commands, noting the output of each one:

```bash
cd /home/linux/ieng6/ln165f/public
ls
cd ~
cd ../public
pwd
ls
```

• What is the output of the fifth command? What does it mean?
  **Solution:** The output is
  ```bash
  /home/linux/ieng6/ln165f/public
  ```
  which is the directory that you’re in after executing these five commands.

• Compare the outputs from the second and sixth commands. Why do you see the similarities and/or differences between the outputs?
  **Solution:** Both commands give the output

  ```bash
  R bin include prefixparser-2004-11-30
  RCS broadcast lib retroweaver-2.0.7
  README.class broadcast.sh modulefiles share
  README.instructor build nltk_data
  ```

  because you’re actually in the same directory at both points which can be verified by executing `pwd` after the first command.

  **[Hint: you can get information about any command by executing the command `man <xxx>`, where `<xxx>` should be replaced by the name of the command.]**
2. Complete problems 2, 10, and 16 from Chapter 1 of the NLTK book.

**Solution:**

**Problem 2:** The answer is $26^{100}$, or

$$31429306415829388301743577885016264272826699887624752563741731753989959$$

$$08420104023465432599069702289330964075081611719197835869803511992549376L$$

(YES, A LONG NUMBER!).

**Problem 10:**

```python
>>> my_sent = ["The","quick","brown","fox","jumped","over","the","lazy","dog"]
>>> x = ' '.join(my_sent)
>>> x
'The quick brown fox jumped over the lazy dog'
>>> x.split()
['The', 'quick', 'brown', 'fox', 'jumped', 'over', 'the', 'lazy', 'dog']
```

Note that it doesn’t matter if you mix '' and "" delimiters for different strings in the same list.

**Problem 16:**

```python
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(10,20)
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
>>> range(10,20,2)
[10, 12, 14, 16, 18]
>>> range(20,10,-2)
[20, 18, 16, 14, 12]
```

Note that some of you may have mistyped the last command the first time around and gotten a different answer:

```python
>>> range(20,10,2)
[]
```

Think about why this is the answer that you get for this command invocation.
3. Go to [http://translate.google.com](http://translate.google.com) and try using Google’s automatic translation system. What kinds of sentences are easy for Google’s system to handle, and what kinds of sentences are hard? See if you can figure out how to craft a sentence that Google’s translation service really botches. What are the properties of that sentence that seem to give the service a hard time? Illustrate using examples of original and translated sentences.

If you’re proficient in a foreign language that Google translates to/from English, you can do this assignment using English paired with that language. If not, then you can do this assignment using “round-trip” translation: write English sentences, translate them into another language, then translate the resulting sentence back into English, and compare the original and resulting sentences.

**Solution:** I’ll use round-trip translation from English to Chinese to English as an example; this is a language pair that Google has no doubt invested a great time in. Here is a case I find pretty impressive:

I didn’t see no stinking traffic light! → I do not see any stinking traffic lights!

I am a bit disappointed at the loss of the past tense in the round trip, but the proper interpretation of the double negation (i.e. semantic interpretation as single negation rather than double) is pretty impressive.

Here is a case where the round-trip translation is successful, but hides a bad intermediate translation:

Each person in the room speaks two languages. →

每个人在房间里讲两个语言。 →

every person at room inside speak two language.

Everyone in the room speaks two languages.

The problem here is that the Chinese translation isn’t a legitimate way to express the meaning; in Chinese, a phrase like 在房间里 (“in the room”) should appear before the noun it modifies, not after. But there’s still enough information in the botched translation to save the round trip back to English. So round-trip evaluation can hide bad problems sometimes.

Here are some totally botched translations:

Mary and Susan date Adam and Paul respectively → Dates of Mary and Susan Adam and Paul

The reason for this is that the interaction of the structure of the sentence with the word *respectively* is quite complex and very particular to English. Chinese doesn’t have a
word like *respectively*, and would need to resort to a totally different sentence structure to get the same meaning across. Since machine translation is still no good at remapping to dramatically different global sentence structure while preserving meaning, Google botches this one.

I went to the city where I always wanted to go and all I got this was this crummy T-shirt. → I went to the city, I’ve always wanted and I went to, it is this lousy T-shirt.

English-to-Chinese translation still doesn’t handle relative clauses (*where I always wanted to go*) well. However, round-trip translation through some other languages (e.g., Greek) works better for this example.