Solve problems A-C in a single Python file `regexp.py`. Your code should include comments to explain any obscure or tricky bits. It should also include demonstration code. Hand in `regexp.py` electronically, either using `hsp` or by dropping the file in your hand-in folder on the COURSES file server. This work is due Monday 2009 April 20 11:59 PM.

Solve problem D on paper, and hand it in Monday in class.

A. Come up with a Python regular expression that describes mis-capitalized words. For the sake of this problem, a word is said to be mis-capitalized if it consists of two or more letters and any letter after the first one is upper-case. You may assume that only alphabetical characters appear in words. For example, when I feed your regular expression to

```
re.findall(yourregexp, ’This is okay tHis IS nOT Okay.’)
```

I should get a result of `[’tHis’, ’IS’, ’nOT’]`.

B. In this problem we learn how to harvest e-mail addresses from texts such as web pages. You must promise never to use this power for evil.

An e-mail address such as `jdavis@carleton.edu` consists of a local part, `jdavis`, and a hostname, `carleton.edu`. The local part is a string made of one or more characters from this set: upper- and lower-case English letters, the digits `0` through `9`, the characters `!`, `#`, `$`, `%`, `&`, `’`, `*`, `+`, `-`, `/`, `=`, `?`, `^`, `_`, `‘`, `{`, `|`, `}`, `~`, and the period `.`. The period is allowed to be neither the first nor the last character in the local part. The hostname is a string made of one or more characters from this set: lower-case English letters, the digits `0` through `9`, the period `.`, and the hyphen `-`. The local part and the hostname are separated by a single `@`. (There are a few more rules to real e-mail addresses, but this is good enough for our purposes.)

Design a Python regular expression that matches e-mail addresses as just specified.

C. I like to write my dates in the format `yyyy/mm/dd`, but sometimes I accidentally write `mm/dd/yyyy` because that’s how I was raised. Write a Python function `fixdates` that takes a string as input, uses a regular expression to fix all dates in the string to my liking, and outputs the fixed string. You may assume that all years are four-digit — I’m Y2K-compliant — but remember that months and days can be one- or two-digit. (Hint: You need to use substitutions and multiple groups.)

D. Prove that Python’s regular expressions are actually more powerful than the regular expressions defined in textbooks. (Hint: You just have to find one problem solvable by Python regular expressions that is not solvable by textbook regular expressions. Hint: Use a group.)