6. (40 points, with 10 bonus points; Exercise 13.22, much extended) Text categorization is the task of assigning a given document to one of a fixed set of categories on the basis of the text it contains. Naive Bayes models are often used for this task. In these models, the query variable is the document category, and the "effect" variables are the presence or absence of each word in the language; the assumption is that words occur independently in documents, with frequencies determined by the document category. In this exercise, we will use Naive Bayes to construct a spam classifier for SMSs, where the two document categories are *spam* and *ham* (i.e., not spam).

   (a) Explain precisely how such a model can be constructed, given as "training data" a set of documents that have been assigned to categories.

(b) Explain precisely how to categorize a new document.

(c) Is the conditional independence assumption reasonable? Discuss.

(d) Write a program to classify SMSs using the data in `sms_spam.zip` (available on Carmen) and measure its accuracy on the test set. This data is from the UCI machine learning repository; see the readme in the zip file for details, and see below for further instructions.

(e) (bonus) A drawback of using relative frequency to estimate the likelihood of a vocabulary element (word/token) appearing in a document is that zero counts result in estimates of zero probability. However, just because a word does not occur in a document class in the training data does not mean that it cannot occur in any document of that class. For example, a word that appears in one or more ham messages in the training set, but in no spam messages, may well appear in one of the spam messages in the test set. Why is this a problem? Devise a solution for avoiding zero probabilities in your distributions, and show that this yields higher accuracy on the test set.

Your program should read in the data file and split it into training and test sets, using the first 80% of the items for training and the remaining 20% of the items for testing. To keep things simple, you can strip punctuation from the messages and split them into words (or more generally, tokens) just based on whitespace. A sample Python 3 program for doing this (`spamreader.py`) along with its output (`reader.log`) is given in the zip file.

In addition to computing the accuracy on the test set, your program should show the spam and ham probabilities along with the ensuing classification prediction for each of the first 50 test items.