**S7.2 Write a program to prompt the user for a file name, and then read through the file and look for lines of the form:**
`X-DSPAM-Confidence:  0.8475`

**When you encounter a line that starts with "X-DSPAM-Confidence:" pull apart the line to extract the floating point number on the line. Count these lines and the compute the total of the spam confidence values from these lines. When you reach the end of the file, print out the average spam confidence.**

```
Enter the file name: mbox.txt
Average spam confidence: 0.894128046745
Enter the file name: mbox-short.txt
Average spam confidence: 0.750718518519
```

```
####################
#    Question 1    #
####################

# Reading file
filename = raw_input("Enter the file name: ")
f = open(filename)

# Creating empty list and storing all the required lines
lst = []

for lines in f:
    if 'X-DSPAM-Confidence:' in lines:
        lst.append(lines)

count_of_lines = len(lst)

spam_confidence = []

# Extracting required number from all the lines and storing in a list
for items in lst:
    spam_confidence.append(float(items[len('X-DSPAM-Confidence:'):]))

total_spam_confidence = sum(spam_confidence)

average_spam_confidence = total_spam_confidence/count_of_lines
print "Average spam confidence: ", average_spam_confidence
```

**S8.5 Write a program to read through the mail box data and when you find line that starts with "From", split the line into words using the split function. Parse the From line and print out the second word for each From line and also count the number of From (not From:) lines and print out a count at the end.**

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

**This is a sample good output with a few lines removed:**

```
python fromcount.py
Enter a file name: mbox-short.txt
stephen.marquard@uct.ac.za
louis@media.berkeley.edu
zqian@umich.edu

[...some output removed...]

ray@media.berkeley.edu
cwen@iupui.edu
cwen@iupui.edu
cwen@iupui.edu
There were 27 lines in the file with From as the first word
```

**Do not use the regexp module.**

```
####################
#    Question 2    #
####################

# Reading file
filename = raw_input("Enter the file name: ")
f = open(filename)

# Creating empty list and storing all the words from the appropriate line
lst = []

for lines in f:
    if lines.startswith("From") and (not lines.startswith("From:")):
        lst.append(lines.split())

# Printing 2nd word from all the lines
for items in lst:
    print items[1]

# Printing total count
print "There were %d lines in the file with From as the first word" % len(lst)
```

**S9.3-4 Write a program to read through a mail log, and build a histogram using a dictionary to count how many messages have come from each email address and print the dictionary.**

```
Enter file name: mbox-short.txt
{'gopal.ramasammycook@gmail.com': 1, 'louis@media.berkeley.edu': 3,
'cwen@iupui.edu': 5, 'antranig@caret.cam.ac.uk': 1,
'rjlowe@iupui.edu': 2, 'gsilver@umich.edu': 3,
'david.horwitz@uct.ac.za': 4, 'wagnermr@iupui.edu': 1,
'zqian@umich.edu': 4, 'stephen.marquard@uct.ac.za': 2,
'ray@media.berkeley.edu': 1}
```

**Now add code to the above program to determine who has the most messages in the file. Specifically, after all the data has been read and the dictionary has been created, look through the dictionary using a maximum loop) to find who has the most messages and print how many messages the person has.**

```
Enter a file name: mbox-short.txt
cwen@iupui.edu 5

Enter a file name: mbox.txt
zqian@umich.edu 195
```

```
####################
#    Question 3    #
####################

# Reading file
filename = raw_input("Enter the file name: ")
f = open(filename)

# Creating empty list and storing all the words from the appropriate line
lst = []

for lines in f:
    if lines.startswith("From") and (not lines.startswith("From:")):
        lst.append(lines.split())

# Creating a dictionary and storing the count of each word
count = dict()
for items in lst:
    count[items[1]] = count.get(items[1],0) + 1

print count

# Finding and printing the email id having maximum count
most_messages_count = None
most_messages_email = None

for key, value in count.items():
    if most_messages_count == None or value > most_messages_count:
        most_messages_count = value
        most_messages_email = key

print "\n", most_messages_email, most_messages_count
```

**S10.2 Write a program to count the distribution of the hour of the day for messages . Pull the hour from the "From" line by finding the time string and then splitting that string into parts using the colon character. Once you have accumulated the counts for each hour, print out the counts, one per line, sorted by hour as shown below. Do not use the regexp module.**

**Sample Execution:**

```
python timeofday.py
Enter a file name: mbox-short.txt
04 3
06 1
07 1
```

```
09 2
10 3
11 6
14 1
15 2
16 4
17 2
18 1
19 1
```

```
####################
#    Question 4    #
####################

# Reading file
filename = raw_input("Enter the file name: ")
f = open(filename)

# Creating empty list and storing all the words from the appropriate line
lst = []

for lines in f:
    if lines.startswith("From") and (not lines.startswith("From:")):
        lst.append(lines.split())

# Slicinf the appropriate field to get the time
hour = []
for i in range(len(lst)):
    time = lst[i][5]
    hour.append(time[0:time.find(":")])

# Counting the occurence of each time
count = dict()
for items in hour:
    count[items] = count.get(items,0) + 1

# Printing the count & time in sorted order
for key in sorted(count):
    print key, count[key]
```

**S11.2 Write a program to look for lines of the form**
`New Revision: 39772`

**Extract the number from each of the lines using a regular expression and the findall() method. Compute the average of the numbers and print out the average.**

```
Enter file:mbox.txt
38549.7949721
Enter file:mbox-short.txt
39756.9259259
```

```
####################
#    Question 5    #
####################

import re

# Reading file
filename = raw_input("Enter the file name: ")
f = open(filename)

# Creating empty list and extracting numbers from the appropriate line
lst = []

for lines in f:
    line = lines.rstrip()
    x = re.findall(r'^New Revision: ([0-9.]+)', line)
    if len(x) > 0 :
        lst.extend([float(i) for i in x])

# Calculating sum to calculate average and printing average
final_total = sum(lst)

average = final_total/len(lst)
print average
```