Burt Rosenberg

Problem Set 1

OUT: 1 SEPTEMBER, 1994
DUE: 13 SEPTEMBER, 1994

Goals

To review arrays, files and the handling of characters and text.

Reading Assignment

Read Chapters 1 and 2 from A BOOK ON C.

Programming Assignment

1. Write a program that counts the number of appearances of each letter *a* through *z* in an input text. Print out a table showing the frequency of each letter, that is, the ratio of the letter's occurrences to the total number of letters in the text.

   Upper case letters and lower case letters are not to be distinguished: an *A* is counted the same as an *a*. Any character which is not a letter, for instance, a digit, a space, a punctuation mark, is ignored.

2. A simple cryptographic protocol is called the *Caesar Cipher,* named after the Roman Emperor and inventor of the Caesar Salad. The two correspondents must first agree upon a small integer k. Before the secret message is sent, it is turned into *ciphertext* by shifting each letter forward in the alphabet by k places. When received, the ciphertext is turn back into *plaintext* by shifting each letter backwards by k places. To shift beyond *z*, consider the alphabet as a ring of characters, so that the next letter after *z* is *a*. For instance, the encoding of *sneeze*, for shift value 3, is *vqhhch*.

   A Caesar Cipher is a very easy cipher to break. It is the ease-dropper's task to determine the secret value of k by looking only at the ciphertext. The frequency of letter use in English has a very distinctive shape and is not hidden by this cipher, it is only shifted over. The ease-dropper can determine k by making graphs of the letter frequencies in the ciphertext and in a sample English language text and sliding them across each other until there is a good match. This will give away the correspondence between ciphertext and plaintext letters. Often the ease-dropper must try a few likely possibilities before succeeding.

Use your program on an English language text of your choosing to get sample frequency counts for letters, and use these to decode the following hilariously funny joke:

```
doha kv fvb jhss h ulhy zpnoalk kpuvzhby? h kvfvbaopurolzhdbz!
```

If there are no word breaks, it becomes harder, but longer text gives a more decisive frequency count. Try this important news item.

```
pybio kbcdr opyvu csxkv sddvo dygxs
xdohk czbyz yconx kwocd ydros bvymk
vzycd yppsm oyxol iyxod rozyc dkvco
bfsmo botom dondr owkvv psxkv visxo
sqrdo oxrex nbonk xnosq rdidr oigby
dokps xkvpe bsyec voddo bdydr oboqs
yxkvz ycdwk cdobg sdrdr scmvy csxqp
bywxy gyxmy xcsno becxk wovoc ckxnc
ydroi nsnxk wovoc cdohk csckl yeddg
oxdiw svocx ybdrg ocdyp kecds xtecd
yppyp xkwov occby kn
```

3. EXTRA CREDIT: A Vigenère cipher works as follows: A keyword is chosen, say "ignatz". It is endlessly repeated,

   ```
   ignatzignatzignatz...
   ```

   The text is written underneath the repeating string, and each letter is shifted according the the letter in the string. That is, if "a" falls under "i", then "a" becomes "i", if "b" falls under "i", then "b" becomes "j", if "c" falls under "t", then "c" becomes "v". Write a program that automatically breaks Vigenère ciphers.