# CS205 - LAB1

Computers have historically been associated with decrypting efforts during WWII. Let’s talk a bit about message encryption. For simplicity, we'll only use basic Latin characters (A to Z), and uppercase.

A very simple system for encrypting a message consists in shifting every letter by one position (every A becomes a B, every B becomes a C, and so forth). A (hardly more sophisticated) system consists in shifting letters not by one position, but by *n* positions, with *n* between 1 and 26. These encryption systems, which were used in the antiquity, are easy to break by analyzing letter frequency, as one letter is always replaced with the same letter.

A far more sophisticated system was invented in the 16th century, and resisted attempts at code-breaking until the 19th century. It consists in a combination of shifted alphabets.

You can write 25 different shifted alphabets:

* The alphabet that starts at B and ends up with …YZA
* The alphabet that starts at C and ends up with …ZAB
* and so forth.

Encoding will replace a letter by the letter at the same position in one shifted alphabet; however, not one but several alphabets are going to be used for the message. **Once again we’ll only work with uppercase letters**.

To know which shifted alphabets are going to be used, and in which order, we are going to use a key, for instance “VIGENERE” (the name traditionally given to this encryption algorithm, after one of its inventors, see https://en.wikipedia.org/wiki/Vigenère\_cipher).

Now, suppose that we want to encode “HELLO WORLD”.

H is the 8th letter in the alphabet. As it’s the first letter in the message, we match it to the first letter in the key, V. We’re going to look for the alphabet starting at V. The letter in the message will be encoded with the 8th letter in the alphabet starting at V, which is C.

E is the 5th letter in the alphabet. We’ll associate it with the second letter in the key, I, and replace it with the 5th letter in the alphabet starting with I, which is M; and so forth. When we have reached the end of the key and still have letters to encode, we restart from the beginning of the key.

In the end HELLO WORLD will become

CMRPB AFVGL

(and you notice that the three Ls are all replaced by different letters).

a) First in your program initialize the array defined as

char alphabets[26][26];

that contains the regular alphabet plus the 25 shifted alphabets (this array isn’t absolutely necessary but it makes some computations in the code easier).

**IMPORTANT HINT:**

You are reminded that in C letters and integers intermix. The (integer) position of a letter in the alphabet can be computed by (if ‘A’ is at position 0)

letter – 'A'

which is interpreted as “code of the letter minus code of ‘A’” (letter codes are consecutive); thus,

'E' – 'A' yields 4

and

'A' + 10 (when considered as a char) becomes ‘K’

b) Use the array above to encrypt messages read from the standard input.

Required interface:

* The password is passed on the command line. If the password contains spaces (which will be ignored by the program) it must be quoted on the command line.
* The text to encode is read from the standard input. It allows reading it from a file.
* The encoded text is written to the standard output.

**Examples** (the program is called here "vigenere" but you can give it any name you want)

vigenere hello < to\_encode.txt > encoded.txt

vigenere "hello world" < to encode.txt > encoded.txt

Any character that isn't a to z or A to Z is ignored. Spaces are ignored in the password, and everything is transformed to uppercase ("hello world" = HELLOWORLD).

Spaces in the text to encode are kept and output without any change. Letters are converted to uppercase. Other characters than basic letters and spaces (even digits) are discarded.

The program mustn't crash if no password is provided!

**Extra-credit**

a) Try to provide encryption without using the alphabets matrix.

and/or

b) Provide decryption