CECS 285: Lab 10 v2

OBJECTIVES:

- Strengthen the basic symmetrical XOR encryption from Lab 9 with a multi-byte key

The theory behind this lab is that the weakness of applying the XOR logic operation on each character with a single key value can be strengthened by rotating through multiple key values.

For example if we have an array of 4 key values defined as:

```
keyvals:   db   0x23, 0xff, 0xaa, 0x3c, null
```

And plain text message:

'Go Beach!'

Then we will be able to cycle through the key values and XOR each plain text character with a different byte of the keyvals array. Once the null character of the keyvals array is encountered, the first byte will be used to encrypt the next character. The table below shows how each plain text character would be associated with each byte of the keyvals array:

Plain text character	keyvals byte
G	0x23
o	0xff
	0xaa
B	0x3c
e	0x23
a	0xff
c	0xaa
h	0x3c

CECS 285: Lab 10 v2

ACTIVITY 1: Get the mostly completed source file from beachboard for this lab. The file can be found in the Content Section under the Lab10_Files sub section as a pdf named Lab10_act1. In this file much of source code from Lab 9 is shown with a few alterations and there are sections of the program where comments are provided but instructions are missing. Portions of the program are encapsulated with comments named as "begin section from lab 9" and "end section from lab 9", instructions that are listed between these labels need no modification. Your task is to fill in the missing instructions after the "end section from lab 9" comment.

```
;In the following section load the key bytes from ROM into RAM
    mov     r0,             #keyBytesRAMaddress ;initialize RAM pointer
    mov     dptr,           #keyvals2           ;initialize ROM pointer
    mov     keyvalIndex,    #0x00               ;initialize keyvalIndex
LoadKey:
    push    keyvalIndex                         ;preserve keyvalIndex v
    Write missing instruction                   ;load byte of key from
    cjne    a, #0x00,       notNull             ;check for null termina
        jmp LoadDone                            ;if null is found, ente
    notNull:
        mov     @r0,        a                   ;put byte of key into r
    Write missing instruction                   ;restore keyvalIndex va
    Write missing instruction                   ;increment keyvalIndex
    Write missing instruction                   ;increment RAM pointer
    Write missing instruction                   ;continue the loop

LoadDone:
    mov     @r0, #0x00                          ;append null char to st
    Write missing instruction                   ;re-initialize RAM poin

;----------------   END of Initialization/configuration ---------------
```

There are instructions within the main loop that must be filled in as well, carefully consider what is being accomplished in each section of the program while Writing missing instructions. When your program is complete then you are ready to test. Make a hex file, load the program onto your 8051 and test the ability to perform symmetrical encryption with this

CECS 285: Lab 10 v2

ACTIVITY 2: Repeat the process outlined in Lab 9 Activity3 using:

- a plain text message of your choosing which must be at least 30 characters long and must be a legitimate message i.e. song, quote, statement, haiku, … not random characters
- a **keyvals** array containing at least 10 bytes of your choosing:
    - as was demonstrated in the lab example, the keyvals can be a string of ascii characters or individually defined hex values

Take screenshots of the following steps in the process:

- a screenshot showing the contents of your plain text file with NUL character
- a screenshot showing the encrypted message displayed in Hyperterminal
- a screenshot showing the contents of your cipher text file with NUL character

Deliverables:

- Completed program source from Activity 2 including the keyvals array you have defined
- The screenshots specified in Activity 2
- Manual verification that the first 4 characters of your plain text message were encrypted correctly with your keyvals. To represent this manual verification:
    - Convert each of the first four ascii characters to hex ascii encoded value
    - Convert the raw hex of the first four characters to binary
    - Exclusive Or each binary quantity with your chosen key to produce the binary encrypted value that represents the encrypted character
    - Convert the binary encrypted value that represents the encrypted character to hex
    - Convert the hex value that represents the encrypted value to its corresponding ascii character

There is no demo portion for this lab