

## NWEN 243 Lab 1 - A Simple Cypher (2.5%)

Marked by demonstration in your scheduled lab, 24 July – 4 August 2017

---

### Experience thinking about security

- Write a small program to encode a block of plain-text using a modified Caesar cypher.
- Write a small program to decode a block of cypher-text using a modified Caesar cypher.

### Requirements

- This lab an individual lab written in C.
- We will be writing programs that you execute from the shell command line.
- You must demonstrate during your scheduled lab session in the week: 20-24<sup>th</sup> July.
- You are to write two small C programs that will
  1. encode using your key, and
  2. decode using the same key –using a **modified Caesar cypher**.

For example

```
%>cat file.txt | encode password | decode password > newfile.txt
// I should then be able to execute diff, and see no changes, e.g.:

%> diff -i file.txt newfile.txt
```

### The Modified Cypher

The modified Caesar you will be implementing is a combination of Caesar cypher I and II.

Caesar cypher I used an offset alphabet, where the mapping between plain and enciphered characters was determined by a numerical offset, in the following example 13, so A maps to M:

Plain	A	B	C	D	E	F	G	H	...
Enciphered	M	N	O	P	Q	R	S	T	...

The Caesar cypher II uses a 'key word' where it is placed at the start of the encipher mapping – sans duplicates, the key in this example is "I came, I saw" (without punctuation or whitespace etc.):

Plain	A	B	C	D	E	F	G	H	...
Enciphered	I	C	A	M	E	S	W	X	...

The next letter in the encipher mapping is the next non duplicate letter of the alphabet – in this case X follows W, and it is not a duplicate. This is followed by Y, Z, A, B, ... Note A is not used as it is a duplicate.

**In our modified cypher**, what we will do is use an offset (like cypher I) which will be the length of the key (including duplicates and whitespace and punctuation, i.e., the key "I came, I saw" has a total length of 13. From this offset of 13, we will then apply a normal Caesar cypher II, as in the following example:

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
H	J	K	L	N	O	P	Q	R	T	U	V	I	C	A	M	E	S	W	X	Y	Z	B	D	F	G

Note: numbers included only for your reference

So, 'Hello' – would encode to: 'QNVVA'

## Assumptions

You may assume text is in ASCII and that you do not need to preserve case (just make it all upper case – note ‘-i’ option to diff above) encode symbols, punctuation, spaces or whitespace in general (i.e just pass it through without change), e.g.: “don’t do it” becomes “efh’w ef xw”

## Skeleton Code

A small skeleton will be provided on the course web page to ease you into this project. No IDE is provided, the command line and any editor (vi, emacs, kwrite) will be sufficient. You can compile your code using the command line:

```
%> gcc encode.c -o encode
```

## Submission & Marking

- This project is marked by demo. It needs to be shown to your tutor by the end of your scheduled lab sessions during [24 July – 4 August 2017](#). You may demo your code in the earlier lab session if complete.
- A successful demonstration will award the full 2.5% for this project.
- You should expect to explain briefly your code to your tutor.
- If you have not successfully demonstrated your work by the end of your scheduled lab sessions during 24 July – 4 August 2017, your tutor may award partial marks.