CST8234 – C Programming
Assignment 03A: Encryption Algorithms

> Cryptography or cryptology; from Greek κρυπτός kryptós, "hidden, secret"; and γράφειν graphein, "writing", or -λογία-logia, "study", respectively is the practice and study of techniques for secure communication in the presence of third parties.
>
> Wikipedia

In this assignment, you are going to be working with a **set** of encryption algorithms. The problems are independent, but they need to be solve in the order indicated.

You are going to be working with three different types of encryption algorithms, I'm presenting the first one in this document, once you have written the first one, you will use that to decrypt a second document that contains the information about the second encryption algorithm. With the second algorithm and a new giving key, you will decrypt the third and last problem.

Marking scheme:

Encryption Algorithm	Marks	Required
Basic Encryption	3%	YES
Second algorithm	3%	NO
Third algorithm	2%	NO

This assignment is worth 6% of your final mark. If you decide to write just the first part, you will get 50% in the assignment, but I will consider it as a successful submission. If you decide to go ahead and write the second part, you will be marked with a 100%, and if you decide to go ahead and do the third part, that will be extra marks.

Basic Encryption Algonquin

In this encryption algorithm, each byte of *plaintext* (unencrypted) data has been adjusted arithmetically by adding an "encryption key" value to the byte. An 8-bit byte of data can have values that range from 0-255 (expressing the 256 8-bit patterns from 00000000 to 11111111).

To produce the *ciphertext*, the encryption key value is added to each *plaintext* byte. Where the resulting sum exceeds the maximum value allowed in a byte (the sum is greater than 255), the newly calculated number will be "wrapped around" to the start of the range of available numbers.

The following chart shows a sample of several *plaintext* data byte values going through the conversion to *ciphertext* using an example encryption key of 5. For example, the *plaintext* letter 'A' (byte value 65) has the encryption key 5 added to it to become the *ciphertext* letter 'F' (byte value 70).

Note that even if the unencrypted *plaintext* input data is all printable characters, the resulting *ciphertext* will likely contain many data byte values that are not printable characters. For example, a *plaintext* letter 'A' encrypted using a key of 207 would generate a *ciphertext* byte of 65+207-256=16 and 16 does not correspond to any standard printable character. (Some operating systems assign private non-standard printable glyphs to every character value; so, you may or may not see anything print on your screen for such *ciphertext* bytes.)

Plaintext Character	*Plaintext* Byte Value	Key	*Ciphertext* Byte Value	*Ciphertext* Character
A	65	5	70	F
B	66	5	71	G
C	67	5	72	H
a	97	5	102	f
b	98	5	103	g
c	99	5	104	h
z	122	5	127	⌂
space	32	5	37	%
1	49	5	54	6
2	50	5	55	7
3	51	5	56	8
þ	254	5	3	♥
ÿ	255	5	4	♦

You are to write a small C program `cipher`, to encrypt / decrypt a file. The program should have the following functionality:

```
cipher [ OPTIONS ] SOURCE DESTINATION
```

`OPTIONS:`

`-d` KEY

decrypt the file SOURCE using KEY and writes back into DESTINATION

`-e` KEY

encrypt the file SOURCE using KEY and writes back into DESTINATION

`-h`: help in using the command

Option `d` and `e` are exclusive, if `d` is on, `e` can not be on, and both of them require an argument, the encryption KEY

In case that the user does not give all the required command arguments, you program should print a usage message and exit with EXIT_FAILURE.

Assignment Progression

Now that you have finished your first algorithm, use your **`cipher`** and the KEY **`8234`**, to **`decrypt`** the file **`mystery01`** file that you'll find in BB.