**Mini Project #2. Caesar Cipher** | **Due date: 4 January, 2017**
**Fundamentals of Programming** | **Total points 20**

# Background

Cryptography is the study to convert or encode the text so that it is unintelligible to the common people, but the designated receiver can use an algorithm to easily recover the original text. The original text is called plain-text, while the encoded (or encrypted) text is called cipher text.

# Caesar cipher

Caesar cipher is an old easy algorithm used in the history by the Roman emperor Julius Caesar. Caesar cipher uses a fixed key $k$, with $1 \leq k \leq 25$, and every character of the plain-text is shifted by the fixed key $k$ to a new character. For example, if $k = 2$, then ’a’ -> ’c’, ’b’ -> ’d’, ’c’ -> ’e’ etc. (and also the upper case characters ’A’ -> ’C’, ’F’ -> ’H’ etc.). To decipher the cipher text, you just use $-k$ or the inverse of the key $k$ with $-1 \geq -k \geq -25$ to shift (and wrap) to decipher if you know what key $k$ was used to encrypt. In case you don’t know what key $k$ was used but you know Caesar cipher was used to encrypt, then you can (by brute force) manually try all combinations of $-1, -2, \cdots,$ up to $-25$ (or try programmatically a for loop to decipher).

Note that at the end, the characters are wrapped around. Hence ’Z’ + 1 = ’A’ here (although from the ASCII table, ’Z’ + 1 = ’’.

Caesar cipher is very easy to comprehend. However, manually shift a long string by $k$ to encrypt and manually shift by ˘$k$ to decipher and also wrap around at the end is not trivial.

## Cryptanalysis of Caesar Cipher

Caesar cipher is readable for the receiver with a key or with enough patience (manually) or with a program. For a stranger like a cracker who intercepts the message, it is readable by trying all 25 key combinations. This is not secure at all.

## Task #1. Encrypt

Write a function encrypt() to read a string input, a key $k$, $1 \leq k \leq 25$ and then encrypt that. The string can consist of upper case and lower case characters. Both upper case and lower case characters will be shifted and then wrapped around. The string can contain special characters such as ’ ’ (space), ’,’ (comma), ’.’ (period) etc., and your code will not shift the special characters.

As a way to verify your program, with the input "Rome is not built in one day.", the output will be “Urph lv qrw exlow lq rqh gdb.”

## Task #2. Decrypt

Write a function decrypt() to decipher a string encrypted by the encryption program from Task #1. The program will be simple since you just make key values $k$ with $-1 \geq k \geq -25$ instead of between 1 and 25 to decipher. Given a string like "Urph lv qrw exlow lq rqh gdb." from Task #1, you can decipher using the same code of Task 1 if you know key $k = 3$ was used to encrypt (then you use $k = -3$ to decipher).

You can decipher even if you don’t know the key used in Task #1 to encrypt by just using a for loop iterating through $k = -1$ to $k = -25$ to try out all the possible keys.

Try to decipher this string "Enwr, Ermr, Erlr."

## Hand-in

Hand-in ONLY the files listed below at the appropriate location on the blackboard system at LMS. You should hand in a single compressed/archived file named Project_2_<your reg. No. XXX without angle brackets>.zip that contains the following files.

1. COMMENTED source files (caesar.c) with author information at the beginning.
2. CLEAR snapshot of your console (SNAPSHOT.png) to show that your program is working.
3. PLAIN text file (OUTPUT.txt) that includes a) author information at the beginning, b) a brief explanation of the project, c) difficulties faced, and d) any comments, or suggestions.

Programs will be graded on the following criteria:

1. **Program Specifications / Correctness.** does it compile? Are there obvious errors? Are there subtle errors? (25%)
2. **Documentation.** Is your program consistently indented in a manner that reflects the structure of your code? Is it easy to read? Are there blank lines which break up the major sections of your code? (15%)
3. **Efficiency.** Is your program efficiently organized, or is there a lot of duplicated code? Does it look well-written, or barely finished? (5%)
4. **Project Specifications.** Does your program fulfill the basic requirements? Is it done? And what else does it do? (15%)
5. **In time.** (10%)
6. **Plagiarism.** (30%)

## Honor code

The student should agree to the terms below; any infringements will result in minimum marks.

- will not cheat on project
- will not share solutions to the project; and
- will notify the instructor immediately if he or she becomes aware of any other group cheating

## References

Caesar Cipher