

CS 135: Fall 2010. Project 2 – Simple Cryptography

Project Rules: You should work on the project in your assigned team. This project is worth 60 points towards your total projects grade. If you choose to implement the simple version (to provide a simple solution) then you can earn only up to 50 points. If you choose to implement the general version then you can earn up to 60 points. In addition, you can earn extra points based on additional features that you build into the program (if the features are shown to be useful). There is no collaboration, of any sort, allowed between teams on this project. In addition, you must try to complete Team Homeworks 5 and 6 since you will need that code for this project.

Project Description

Cryptography, from the Greek word *kryptos*, meaning "hidden", deals with the science of keeping information secret. The desire to do this has existed ever since humankind was first able to write. There are many ways to keep something secret. One way is to physically hide the document. Another way is to encrypt the text you wish to remain secret. Some people use this to keep others from understanding the contents of the files in their computer. Encrypting a file requires an input message, called the "plain text," and an encryption algorithm. An encryption algorithm transforms "plain text" into "cipher text." Just like a door needs a key to lock and unlock it, an encryption algorithm often requires a key to encrypt and decrypt a message. Just like a homeowner can selectively give the key to the front door to only those he wants to allow unaccompanied access, the author of a message can selectively give the encryption key to only those he wants to be able to read the message. In order for someone to read the encrypted message, they have to decrypt the cipher text, which usually requires the key.

For example, suppose the plain text message is HELLO WORLD. An encryption algorithm consisting of nothing more than replacing each letter with the next letter in the alphabet would produce the cipher text IFMMP XPSME. If someone saw IFMMP XPSME, they would have no idea what it meant (unless, of course, they could break the encryption, or had the key to decrypt it.) The key to encrypt in this case is "pick the next letter in the alphabet," and the decryption key is "pick the previous letter in the alphabet." The decryption algorithm simply replaces each letter with the one before it, and presto: the plain text HELLO WORLD is produced.

Caesar's Cipher Encryption Algorithm

In this programming project you will implement a simple version of a cryptography algorithm called Caesar's Cipher to encrypt your data (the precise data to be encrypted will be described in detail in the next section). The term Caesar's cipher is an ancient encryption technique used by Julius Caesar to send secret messages. At its heart is the concept of modulo arithmetic (recall this from your discrete math, cs123, class and from a homework assignment). Caesar used it to encrypt the alphabet.

Conceptually it works as follows. The input (message) plain text to be encrypted is a string of M symbols $p_1 p_2 p_3 \dots p_M$, each symbol p_i can take on N different values (for example, the English alphabet has 26 letters in uppercase). The other part of the input is the encryption key K which must be less than N . To encrypt the message each character/value p_i in the input is replaced by the cipher text character/value $c_i = (p_i + K) \text{ modulo } N$.

- For example, if we are dealing with the English alphabet (of 26 letters) and the input string is ABCD and the key is 5 then the encrypted string is FGHI – i.e., we are assigning the numeric values 0,1,...,25 to the alphabets A,B,...Z respectively and then computing the sum modulo 26. As another example, if the input is a sequence of digits such as 4738 and the key is 4 (note that the key must be at most 9) then the encrypted sequence is the four digits $(4+4) \bmod 10$, $(7+4) \bmod 10$, $(3+4) \bmod 10$, $(8+4) \bmod 10$ which gives the string 8172.

To decrypt the cipher text we need the key. If the cipher text is a sequence of M symbols $c_1 c_2 \dots c_M$ and the key is K , then to recover the plain text (i.e., to decrypt and recover the original string) we replace each symbol with the plain text symbol $a_i = (c_i - K) \bmod N$. For example, to decrypt the string 8172 with key 4 we replace it with $(8 - 4) \bmod 10$, $(1 - 4) \bmod 10$, $(7 - 4) \bmod 10$, $(2 - 4) \bmod 10$ which gives us 4738.

- Note: Recall from cs123 that $(x-y) \bmod N = (x + y') \bmod N$ where $y' = N - y \pmod N$ (i.e., y' is the inverse of $y \pmod N$). Therefore to decrypt we can add the inverse of the key modulo N . For example, $(1-4) \bmod 10 = (1+6) \bmod 10$ since inverse of 4 is $10-4=6$. You need to use this property to implement your assembly code LC3, so make sure you understand how this works. Note that since the key K is less than N , the number $N-K$ is a positive integer less than N .

Note on applicability of Caesar Cipher: The Caesar cipher is known to be an extremely weak encryption technique. As a result, it is not actually used in any real security applications today.

Project Requirements

In this project you are required to implement a simple version of Caesar's cipher algorithm in LC3 assembly. A complete version is described at the end of this document – this “general” version is really a more generic implementation that should work for numbers, alphabets, and special characters. *You should read the simple version first to get an idea of what is required and then read the general version. Do this before you start coding, to figure out if you want to just implementing this general version first.*

Simple Version:

You are to implement the Caesar's cipher algorithm as described in the previous section to encrypt a sequence of numbers. Applications where a sequence of digits/numbers, such as the social security number or credit card numbers, have to be encrypted are numerous – examples include online purchasing (such as Amazon), patient records in health care

systems. In this assignment, you will develop a program to encrypt 16 digit credit card numbers using Caesar's cipher.

- Note real systems (such as Amazon) do not use Caesar's cipher to encrypt your credit card number – they use more secure techniques such as AES and RSA.

Your encryption/decryption programs (in LC3 assembly) must meet the following requirements. Any deviation from this, without approval from the instructor, or the TAs will result in points being deducted.

Input Interface

Your program should prompt the user for three separate inputs from the keyboard, as follows:

(1) The prompt:

TO ENCRYPT, TYPE E; TO DECRYPT, TYPE D; TO EXIT, TYPE X;

- The user will type E or D. A real world program should also detect any other character typed and respond with
 - THAT IS AN ILLEGAL CHARACTER. PLEASE TRY AGAIN.
- For a lower level of difficulty you can assume in this assignment that the user can type an E or D or X correctly – note that while this will lead to a correct program, if you do not implement the error checking you will lose 2 points.
- Your program will accept that character, store it someplace (you should specify where in memory) and use it, as we shall see momentarily.
- If the user types X then the program exits and halts.

(2) If either E or D is chosen in (1) then the prompt:

ENTER THE ENCRYPTION KEY (A SINGLE DIGIT BETWEEN 1 AND 9).
WHEN DONE PRESS ENTER

- (a) The user will type a single digit, from 1 to 9. Again, we will assume the user can successfully hit digit keys on the keyboard.
 - Question for you to think about: do you want to echo the key that the user has typed in?
- (b) Your program will accept this digit, store it someplace, and use it to encrypt or decrypt the message.
- (c) If the input in (1) was D (decryption) then your program must decrypt the credit card number using the key entered in this step and print it out to the display. The credit card number will be a 16 digit cipher text (i.e., after encryption) stored starting at location with label CARDNUM.
 - Note: to protect against losing the encrypted data stored starting at CARDNUM if a wrong key was entered, make sure you do not write into CARDNUM the string you are going to print.
- (d) The decrypted card number must be printed to the display. Note that if the user entered the wrong key, then it will print out a number that should not be equal to the original number.

(3) If the input in (1) was E (i.e., encryption) then after getting the key in step (2), print the prompt:
INPUT THE CREDIT CARD NUMBER OF 16 DIGITS. WHEN DONE, PRESS
<ENTER>

- The user will input a sequence of 16 digits (recall the LC3 input routine reads in ASCII characters) from the keyboard, terminating the message with the <ENTER> key.
- Your program may store this ASCII string in some location.
- One constraint: The input number must be 16 digits in length; so make sure you check for correct data.
- Using the value of the key entered in step(2), and the encryption algorithm, generate the cipher text and store the cipher text into a location (you will need 16 memory words to store a 16 digit string) labeled with CARDNUM (i.e., the starting address of the location is CARDNUM).

Hint: You will need to use some of the subroutines that you have used or designed before – for example, the Calculator routine to read input strings, check for input command and branch to appropriate routine (or address), and convert from ASCII to binary and binary to ASCII, and the modulo subroutine. (We will post a solution to the Modulo program that you can use.)

General Version:

The Caesar cipher algorithm can be applied to work for any set of symbols (ASCII characters). However, since we have many more values (characters) we cannot do modulo 10 (why?) and must instead pick modulo N, where N is a number large enough to handle all valid symbols. Note that a key is now any number between 0 and N-1 (where N is the size of your set of symbols).

In this general version, you are required to implement the Caesar's cipher encryption algorithm to encrypt (and decrypt) any secret MESSAGE which is a string of valid symbols from the keyboard. Such a string, for example, could correspond to a password stored in the system. As before you can assume a string of 16 symbols. (Optional: you must also determine how to modify this to handle 'at most 16 symbols'?). Assume that the symbols are the symbols on the keyboard – digits, letters, and the 32 special characters. Specifically, assume that you have to deal with all possible symbols starting with “!” and ending at “~”. In terms of their ASCII representation, all characters with ASCII representation from x21 (the symbol “!”) to x7E (the symbol “~”). Your solution must follow a format similar to what was described for the simple version above – and repeated below for your benefit.

Input Interface for General Version of Project

Your program should prompt the user for three separate inputs from the keyboard, as follows:

(1)The prompt:

TO ENCRYPT, TYPE E; TO DECRYPT, TYPE D; TO EXIT, TYPE X:

- The user will type E or D or X. A real world program should also detect any other character typed and respond with
 - THAT IS AN ILLEGAL CHARACTER. PLEASE TRY AGAIN.
- For a lower level of difficulty you can assume in this assignment that the user can type an E or D correctly – note that while this will lead to a correct program, if you do not implement the error checking you will lose 2 points.
- Your program will accept that character, store it someplace (you should specify where in memory) and use it, as we shall see momentarily.
- If the user types X then the program exits and halts.

(2) If either E or D is chosen in (1) then the prompt:

ENTER THE ENCRYPTION KEY (A NUMBER BETWEEN 1 AND N).

WHEN DONE PRESS <ENTER>

- (a) The user will type their key (note that the key in this case is a **number** between 1 and N – *you have to figure out what N is*). For example, if you figure out that the maximum number for the key is 25 then the prompt should say “A number between 1 and 25”. Again, we will assume the user can successfully hit digit keys on the keyboard.
 - a. Question: do you want to echo the key that the user has typed in?
- (b) Your program will accept this number, store it someplace, and use it to encrypt or decrypt the message.
- (c) If the input in (1) was D (decryption) then your program must decrypt the string (some ciphertext/message you stored) using the key entered in this step and print it out to the display. The cipher text string will be a 16 character cipher text (i.e., after encryption) stored starting at location with label MESSAGE.
 - a. Note: to protect against losing the encrypted data stored starting at MESSAGE if a wrong key was entered, make sure you do not write into MESSAGE the string you are going to print.
- (d) The decrypted message must be printed to the display. Note that if the user entered the wrong key, then it will print out a message (string of characters) that should not be equal to the original message.
- (e)

(2) If the input in (1) was E (i.e., encryption) then after getting the key in step (2), print the prompt:

INPUT THE SECRET MESSAGE OF 16 SYMBOLS. WHEN DONE, PRESS <ENTER>

- a. The user will input a sequence of 16 symbols (recall the LC3 input routine reads in ASCII characters) from the keyboard, terminating the message with the <ENTER> key.
- b. Your program may store this ASCII string in some location.

- c. One constraint: The input must be 16 characters in length; so make sure you check for correct data.
- d. Using the value of the key entered in step(2), and the encryption algorithm, generate the cipher text and store the cipher text into a location (you will need 16 memory words to store a 16 character string) labeled with MESSAGE (i.e., the starting address of the location is MESSAGE).

Hint: You will need to use some of the subroutines that you have used or designed before – for example, the Calculator routine to read input strings, check for input command and branch to appropriate routine (or address), and convert from ASCII to binary and binary to ASCII, and the lab assignment to perform modulo arithmetic. (We will post a solution to the Modulo program that you can use.)

Hand-in Requirements – What you need to submit.

You are required to hand-in two components: (1) your assembly code and (2) a report describing your implementation. *You must clearly state which version you implemented – the simple version or the general version.* Your assembly code must be a text file – i.e., an .asm file. You must name the file as <team name>.asm. For example, if my team name was Team 4 then I would submit it as team4.asm. The report is meant to ascertain the effectiveness of your design and we are looking for an algorithm describing your implementation. At the very least, you must document your code, and include a flow-chart describing your solution. Note that you risk a grade of zero if your program does not work and you do not submit a report.