CST8234 – C Programming
Data Cryptography

Setup:

1. Create a directory **`Lastname_05`** . You are going to develop your lab here

Network Cryptography

The explosive growth of Internet communications and data storage on Internet-connected computers has greatly increased privacy concerns. The field of cryptography is concerned with coding data to make it difficult (and hopefully—with the most advanced schemes—impossible) for unauthorized users to read.

In this exercise you'll investigate a simple scheme for encrypting and decrypting data. A company that wants to send data over the Internet has asked you to write a program that will encrypt it so that it may be transmitted more securely. All the data is transmitted as **four-digit integers**.

Your application should read a four-digit integer entered by the user and encrypt it as follows:

- Replace each digit with the result of adding 7 to the digit and getting the remainder after dividing the new value by 10.
- Swap the first digit with the third
- Swap the second digit with the fourth.
- Then print the encrypted integer.

Write a separate application that inputs an encrypted four-digit integer and decrypts it (by reversing the encryption scheme) to form the original number.

Program #1:

Write a small C program **`crypto.c`** that:

1. Ask the user an integer number
2. Reads an integer number
3. Checks that the number is at least **`4`** digits long
4. If the number of more than **`4`** digits, it should print a message to the end user "**`Invalid number of digits`**" and terminate with an exit value of **`EXIT_FAILURE`**
5. If the number of digits is correct, your program should encrypt the number with the encryption algorithm given above.
6. At then end, your program should print the encrypted number and terminate with an value of **`EXIT_SUCCESS`**.

The following demonstrates the execution of the program:

```
root@luna:CST8234/# ./crypto
Enter an  4 integer number:  1256
Encrypted number of 1256 is 2389
root@luna:CST8234/# echo $?
0
root@luna:CST8234/# ./crypto
Enter an  4 integer number:  123456
```

```
Invalid number of digits 6 in 123456
root@luna:CST8234/# echo $?
1

                              SAMPLE TEST OUTPUT:  crypto
```

In order to successfully complete this program and obtain all the marks, you will need to:

1. Use the macros **`EXIT_FAILURE`** and **`EXIT_SUCCESS`** define in the library **`stdlib.h`** to indicate unsuccessful or successful termination of your program
2. Define **`DIGITS`** as a macros in your program. **`DIGITS`** should be defined as **`4`**.
3. Write a function, with function prototype **`int crypto( int )`**
   (A) **`crypto( )`** should encrypt a 4 digit number as establish above.
   (B) Your function should return the encrypted number.
   (C) **HINT**: Separate the digits and store them in an array.
4. Your **`main( )`** should:
   (A) Use the function **`scanf( )`** to read an integer from the keyboard.
   (B) Validate the number entered by the user.
   (C) If the number has more than 4 digits, your program should print an error message and terminate with **`EXIT_FAILURE`**
   (D) If the number has less than 4 digits, your program should invoke a function to encrypt the number.
   (E) Print the newly encrypted number and exit with **`EXIT_SUCCESS`**
5. Your program should be compiled with the flags **`-Wall -ansi -pedantic`**

Program #2:

Copy your **`crypto.c`** to a new program **`decrypt.c.`** Your decrypt program should be able to decrypt a encrypted number.

```
root@luna:CST8234/# ./decrypt
Enter an  4 integer number:  2398
Decrypted number of 2398 is 2156

                              SAMPLE TEST OUTPUT:  decrypt
```

Program #3 & #4:

1. Copy **`crypto.c`** to **`datacrypto.c`** and **`decrypt.c`** to **`datadecrypt.c`**
2. Modify your program **`crypto`** and **`decrypt`** to read from the standard input **`stdin`**
   (A) Use the function **`fscanf( )`** instead of the function **`scanf( )`**
   (B) Read from the **`stdin`** (keyboard) file descriptor
   (C) If **`fscanf( )`** reads the **`EOF`** character, return **`EOF`** otherwise return the number read
   (D) **HINT**: Look at **`readMultiples.c`** posted with this lab.

```
root@luna:CST8234/# ./datacyrpto < 01_data.txt
2134
8910
3125
4756
9018
3245
```

```
0598
```

SAMPLE TEST OUTPUT: datacrypto

Two files **01_data.txt** and **02_data.txt** are provided. Each file contains a number per line. Notice the redirection symbol (**<**) used to call your program using the file as input. You can use this idea to test your program without having to do all the typing! Your lab instructor may use different files to test your program. Be sure that you make some extra files, for example, use negative numbers, wrong numbers at the beginning / end of the file, large amount of data, large numbers, etc