Instituto Politécnico Nacional
Escuela Superior de Cómputo
Cryptography

Practice 1: Shift Cipher

By:
Moreno Zárate Víctor Gibrán

Professor:
M. en C. NIDIA ASUNCIÓN CORTEZ DUARTE
September 2016

Contents

Problem

What is the behavior of a shift cipher?

What are the main characteristics of a shift cipher?

How does the output behave according to the input and the key?

The shift cipher, also known as Caesar cipher, is one of the simplest and most widely known encryption techniques. Its basic principle consists in shifting each letter of a message certain number of positions [1], making the messages hard to read. This is a simple substitution (monoalphabetic) technique.

Hypothesis

A possible solution to the problem is to write a computer program in order to analyze the behavior of the shift cipher, this program will be able to encrypt and decrypt a text using this technique.

If we program this software correctly, we will see how the character data shifts into another one, changing the lowercase into uppercase or vice versa, then we will observe the encrypt/decrypt of the cipher text if we put the correct key.

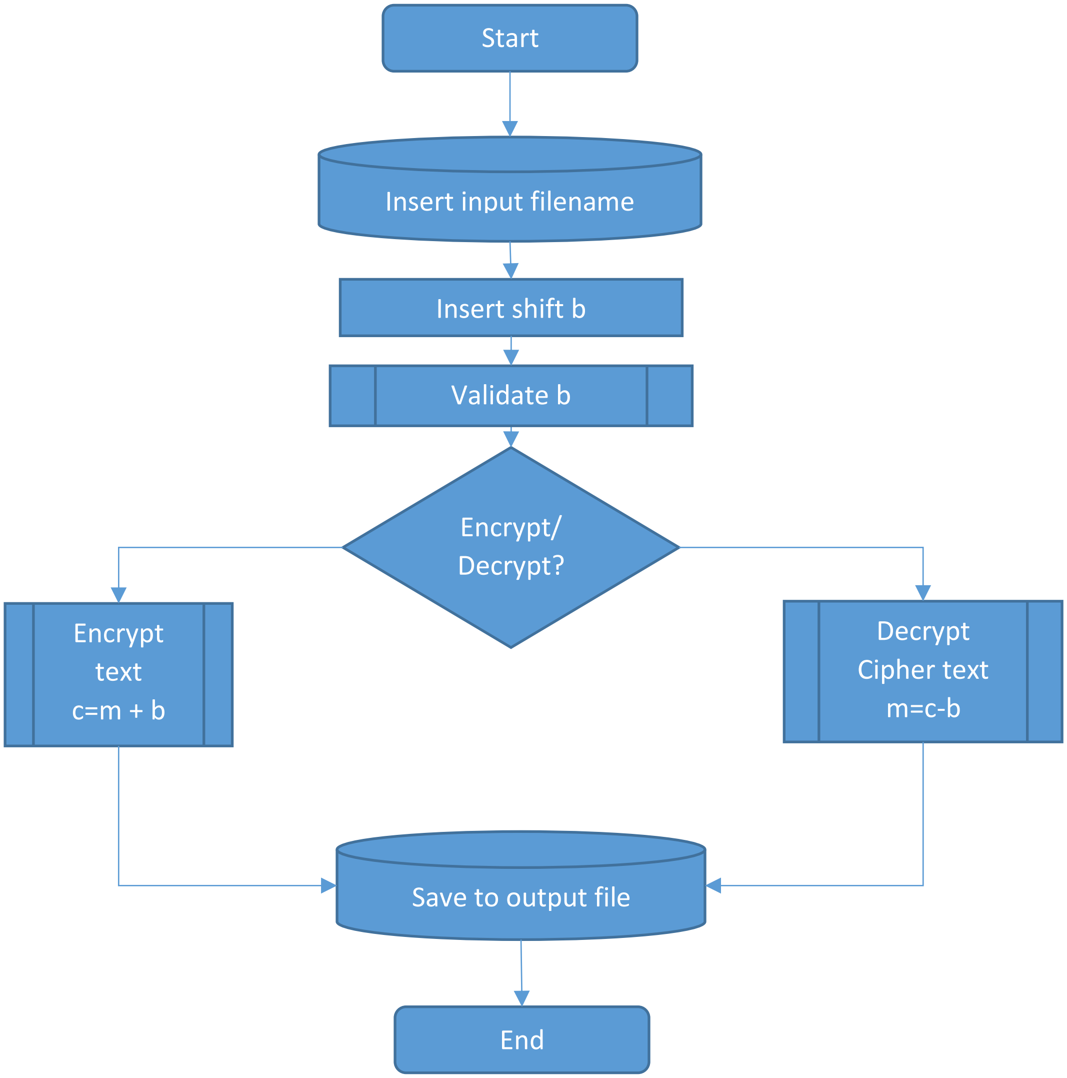
In the case of the bmp image we will see a shift in their colors, but the image shall not change in size or get pixelated.

Software (libraries, package, tools)

In order to do this practice, we used:

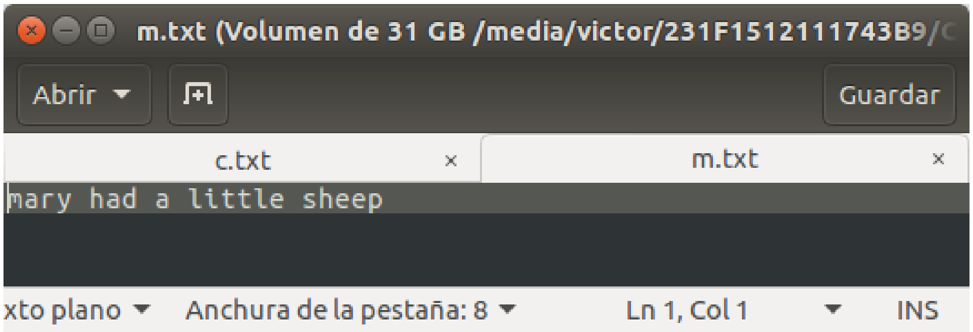
- Personal Computer
- Linux Operating System
- Text Editor
- GNU C Compiler
- Image Viewer
- Image Manipulation Software (GIMP)

Procedure



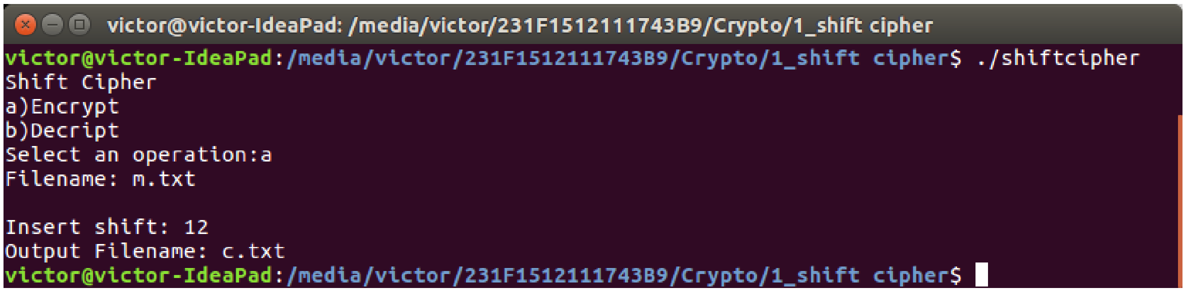
The program asks to the user to input the file name of operation, the shift number to use, and the operation to perform (encrypt/decrypt), then calls the correct procedure and stores the output file with the processed data. It also validates if the file name exists, if it is possible to open/write the file and if the inserted shift is valid (number not 0 or multiple of 26).

Results (Data)



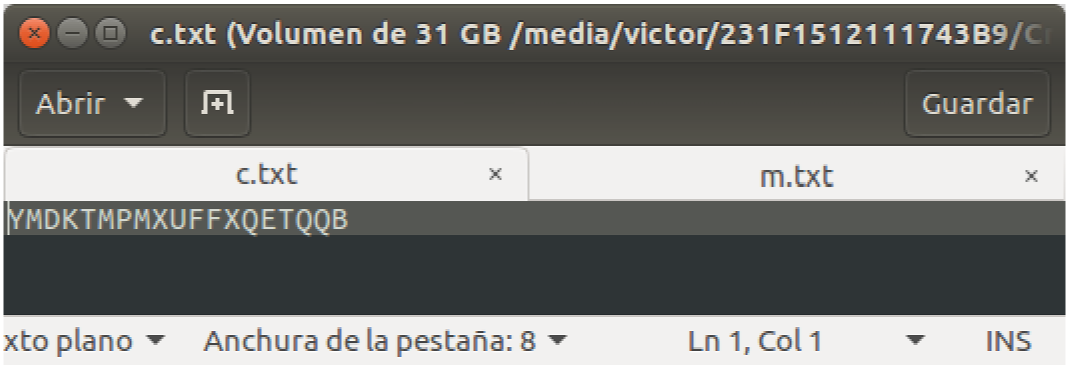
```
mary had a little sheep
```

Figure 1. Image of the original plain text


victor@victor-IdeaPad: /media/victor/231F1512111743B9/Crypto/1_shift cipher

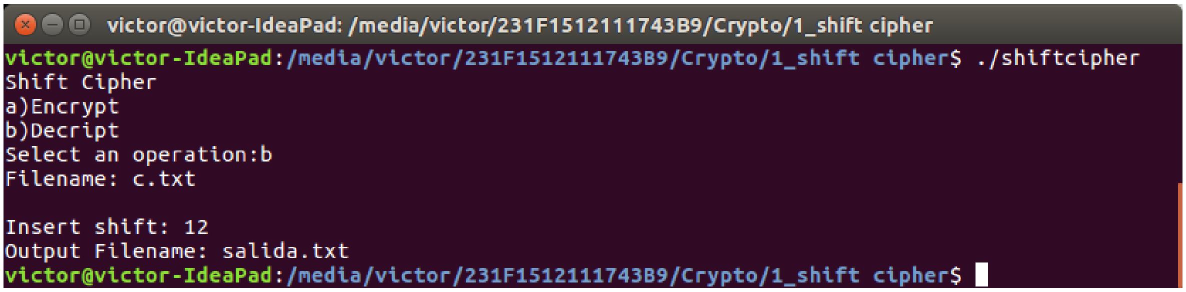
```
victor@victor-IdeaPad:/media/victor/231F1512111743B9/Crypto/1_shift cipher$ ./shiftcipher
Shift Cipher
a)Encrypt
b)Decript
Select an operation:a
Filename: m.txt

Insert shift: 12
Output Filename: c.txt
victor@victor-IdeaPad:/media/victor/231F1512111743B9/Crypto/1_shift cipher$
```

Figure 2: Image of an encrypt operation



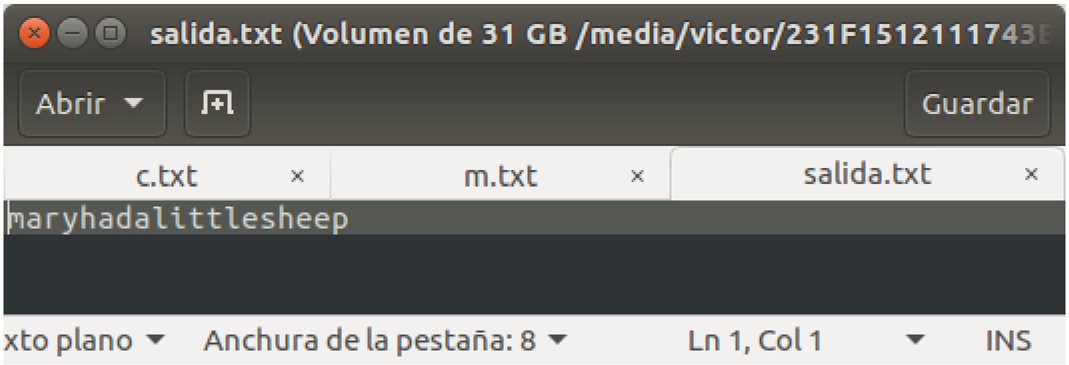
```
YMDKTMPMXUFFXQETQQB
```

Figure 3: Output cipher text


victor@victor-IdeaPad: /media/victor/231F1512111743B9/Crypto/1_shift cipher

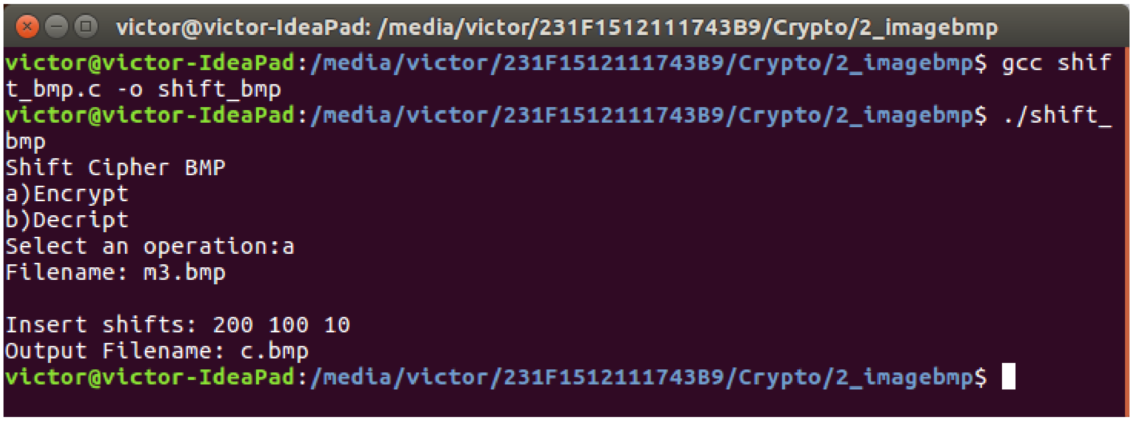
```
victor@victor-IdeaPad:/media/victor/231F1512111743B9/Crypto/1_shift cipher$ ./shiftcipher
Shift Cipher
a)Encrypt
b)Decript
Select an operation:b
Filename: c.txt

Insert shift: 12
Output Filename: salida.txt
victor@victor-IdeaPad:/media/victor/231F1512111743B9/Crypto/1_shift cipher$
```

Figure 4: Image of a decrypt operation



```
maryhadalittlesheep
```

Figure 5: Recovered text.


victor@victor-IdeaPad: /media/victor/231F1512111743B9/Crypto/2_imagebmp

```
victor@victor-IdeaPad:/media/victor/231F1512111743B9/Crypto/2_imagebmp$ gcc shif
t_bmp.c -o shift_bmp
victor@victor-IdeaPad:/media/victor/231F1512111743B9/Crypto/2_imagebmp$ ./shift_
bmp
Shift Cipher BMP
a)Encrypt
b)Decript
Select an operation:a
Filename: m3.bmp

Insert shifts: 200 100 10
Output Filename: c.bmp
victor@victor-IdeaPad:/media/victor/231F1512111743B9/Crypto/2_imagebmp$ 
```

Figure 6: Image of an encrypt operation with a bmp file.

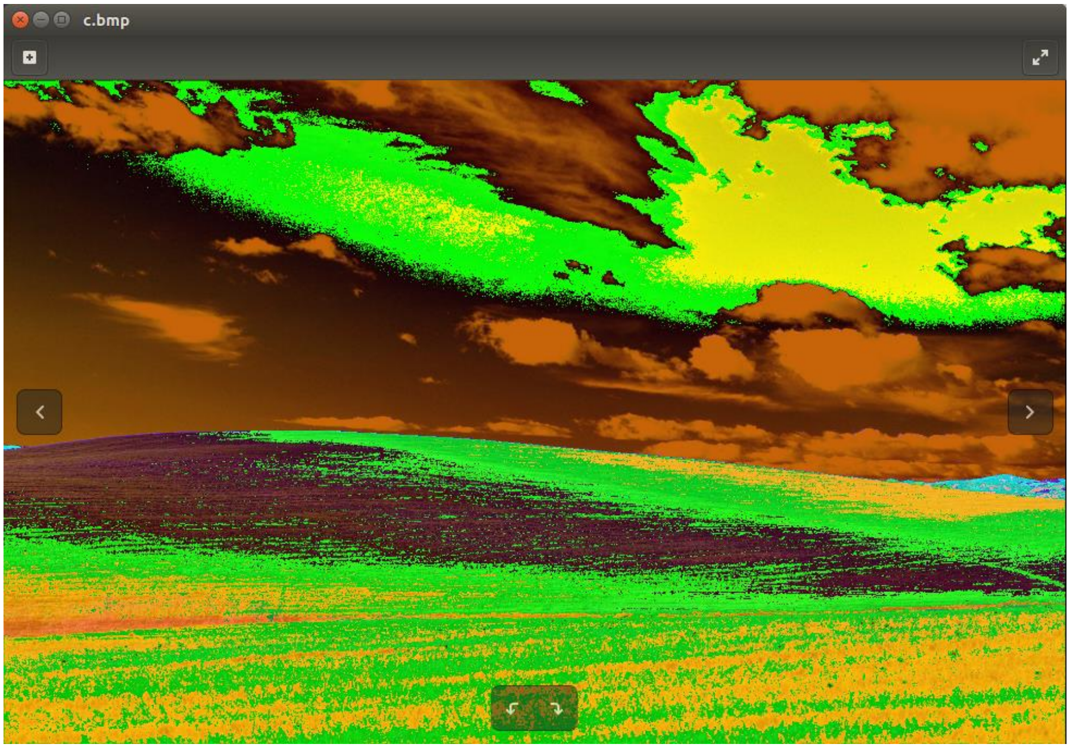
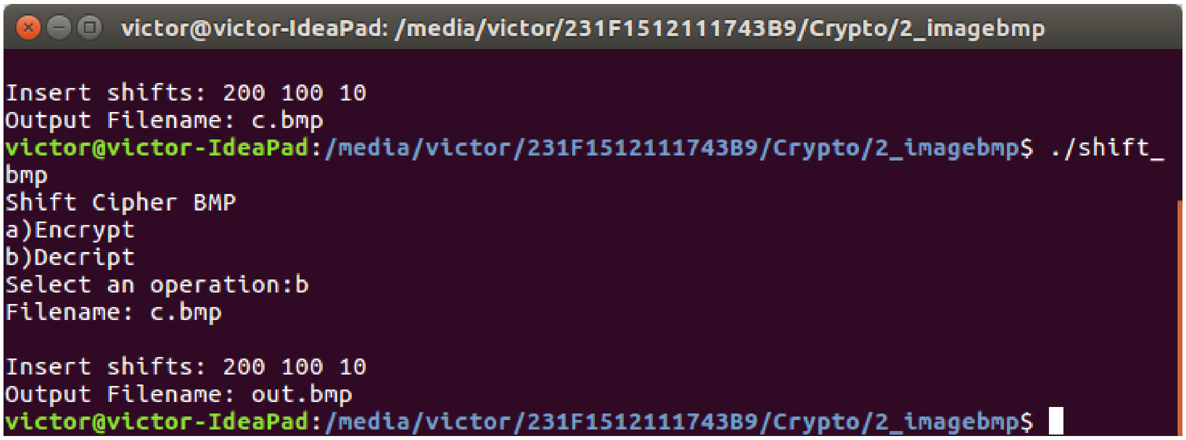


Figure 7: Encrypted image



victor@victor-IdeaPad: /media/victor/231F1512111743B9/Crypto/2_imagebmp

```
Insert shifts: 200 100 10
Output Filename: c.bmp
victor@victor-IdeaPad:/media/victor/231F1512111743B9/Crypto/2_imagebmp$ ./shift_
bmp
Shift Cipher BMP
a)Encrypt
b)Decript
Select an operation:b
Filename: c.bmp

Insert shifts: 200 100 10
Output Filename: out.bmp
victor@victor-IdeaPad:/media/victor/231F1512111743B9/Crypto/2_imagebmp$ 
```

Figure 8: Decryption process.

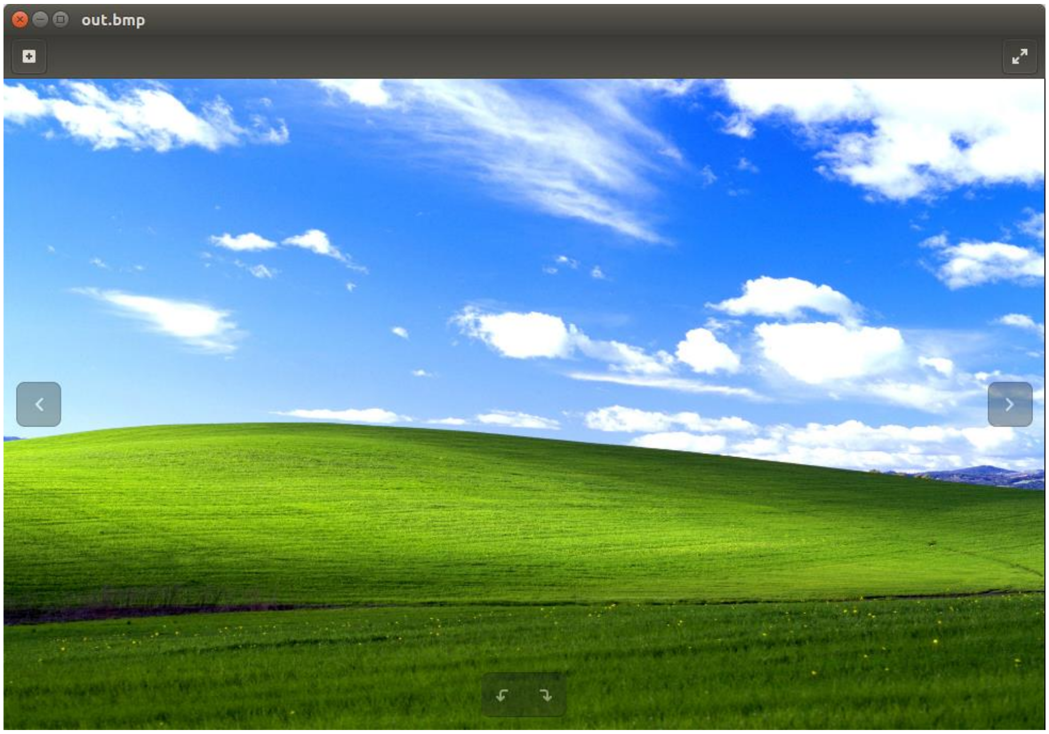


Figure 9: Output Image

Conclusions

I accepted the hypothesis because, according to the observed data of the program executions and the string or colors contained in the output files, I was able to see the expected behavior of the shift cipher, when we put a clear text and encrypt it, and later we decrypt it using the same key, we recovered the message, and if we put extra characters in the cipher text, we observer the monoalphabetic characteristic of this cipher.

I learned the functionality of a basic classical cipher method, in this case the shift cipher, and it can be useful in order to understand more complex methods in this subject, also it can be used in a situation when we need to cipher a simple thing and security is not a “matter of life and death”.

References

[1] "Caesar cipher", *Wikipedia*, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Caesar_cipher . [Accessed: 13- Sep- 2016].

Code

Text

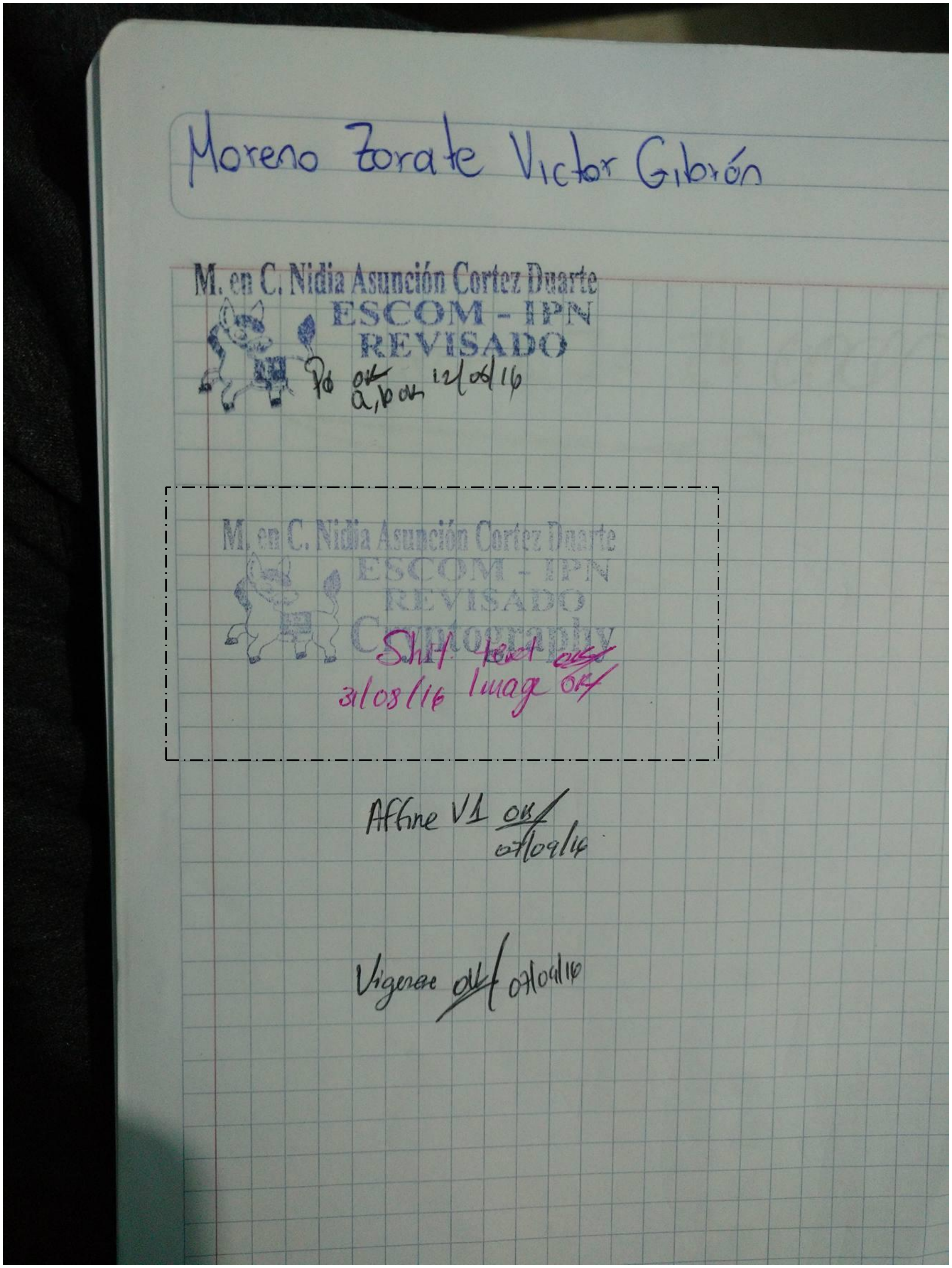
```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
int isnValid(int shift_num);
void io(int shift_num,char filename[256],char output_filename[256],char opt);
void encrypt(FILE *file, FILE *file_dest,int shift_num);
void decrypt(FILE *file, FILE *file_dest,int shift_num);
int mod(int a, int b);

int main(int argc, char *argv[]){
    char opt;

    printf("Shift Cipher\n");//Menu
    printf("a)Encrypt\nb)Decript\n");
    printf("Select an operation:");
    scanf("%c",&opt);

    char filename[256],output_filename[256];
    int shift_num;
    printf("Filename: ");
    scanf("%s",filename);
    do{
        printf("\nInsert shift: ");
        scanf("%d",&shift_num);
    }while(!isnValid(shift_num));

    printf("Output Filename: ");
    scanf("%s",output_filename);

    io(shift_num,filename,output_filename,opt);

    return 0;

    }
//Input/Output function, used to read and write from/to files
void io(int shift_num,char filename[256],char output_filename[256],char opt){
    FILE *file,*file_dest;
    file = fopen(filename, "r");
    file_dest= fopen(output_filename,"w");

    if (file==NULL){
        perror("Can't open file");
        exit(0);
        }

    if (file_dest==NULL){
        perror("Can't create file");
        exit(0);
        }

    if(opt=='a')
        encrypt(file,file_dest,shift_num%26);
    else if (opt=='b')
        decrypt(file,file_dest,shift_num%26);
    else
        printf("Invalid option");

    fclose(file);
    fclose(file_dest);



}
//Encrypt function
void encrypt(FILE *file, FILE *file_dest,int shift_num){
    int caracter,aux;
    while((caracter=fgetc(file))!=EOF){
            if(!isblank(caracter)&&!isspace(caracter)){
                aux=caracter-97;
                aux=mod((aux+shift_num),26);
                aux+=65;
                fputc(aux,file_dest);
                }
        }

    }
//Decrypt function
void decrypt(FILE *file, FILE *file_dest,int shift_num){
    int caracter,aux;
    while((caracter=fgetc(file))!=EOF){
            if(!isblank(caracter)&&!isspace(caracter)){
                aux=caracter-65;
                aux=mod((aux-shift_num),26);
                aux+=97;
                fputc(aux,file_dest);
                }
        }
    }


//Validation of shift number
int isnValid(int shift_num){
    if (shift_num==0){
        printf("Number can't be 0");
        return 0;}
    else if ((shift_num%26)==0){
        printf("Number can't be a multiple of 26");
        return 0;}
    else
       return 1;

      }

      //Definition of mod function
      int mod(int a, int b){
          int r = a % b;
          return r < 0 ? r + b : r;
      }
```

Image

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
int isnValid(int shift_num);
void io(char filename[255],char output_filename[255],char opt,unsigned char RGB[3]);
void encrypt(FILE *file, FILE *file_dest,unsigned char RGB[3]);
void decrypt(FILE *file, FILE *file_dest,unsigned char  RGB[3]);
void head_handler(FILE *file,FILE *file_dest);
void skip(FILE *file,FILE *file_dest,int n);


int main(int argc, char *argv[]){
    char opt;
    printf("Shift Cipher BMP\n");
    printf("a)Encrypt\nb)Decript\n");
    printf("Select an operation:");
    scanf("%c",&opt);

    char filename[256],output_filename[256];
    unsigned char RGB[3];
    printf("Filename: ");
    scanf("%s",filename);

        printf("\nInsert shifts: ");
        scanf("%hhu %hhu %hhu",&RGB[2],&RGB[1],&RGB[0]);


    printf("Output Filename: ");
    scanf("%s",output_filename);

    io(filename,output_filename,opt,RGB);

    return 0;

    }

void io(char filename[256],char output_filename[256],char opt,unsigned char RGB[3]){
    FILE *file,*file_dest;
    file = fopen(filename, "r");
    file_dest= fopen(output_filename,"w");

    if (file==NULL){
        perror("Can't open file");
        exit(0);
        }

    if (file_dest==NULL){
        perror("Can't create file");
        exit(0);
        }

    if(opt=='a')
        encrypt(file,file_dest,RGB);
    else if (opt=='b')
```

```
        decrypt(file,file_dest,RGB);
    else
        printf("Invalid option");

    fclose(file);
    fclose(file_dest);



}

void encrypt(FILE *file, FILE *file_dest,unsigned char RGB[3]){
    head_handler(file,file_dest);

    int caracter=0;
    unsigned char aux=0;
    int num=0;

    while((caracter=fgetc(file))!=EOF){
            aux=(unsigned char)caracter;
            fputc(aux+RGB[num%3],file_dest);
            ++num;
        }

    }

void decrypt(FILE *file, FILE *file_dest,unsigned char RGB[3]){
    head_handler(file,file_dest);

    int caracter=0;
    unsigned char aux=0;
    int num=0;

    while((caracter=fgetc(file))!=EOF){
            aux=(unsigned char)caracter;
            fputc(aux-RGB[num%3],file_dest);
            ++num;
        }




    }



      int isnValid(int shift_num){
          if (shift_num==0){
              printf("Number can't be 0");
              return 0;}
          else
              return 1;

      }

      //Manages header of BMP
      void head_handler(FILE *file,FILE *file_dest){
          int offset;
          skip(file,file_dest,10);
          fread(&offset,sizeof(int),1,file );
          fwrite(&offset,sizeof(int),1,file_dest);
        skip(file,file_dest,offset-14);
        }

    //Skips uncrypted content
    void skip(FILE *file,FILE *file_dest,int n){
        unsigned char caracter;
        int i;
        for(i=0;i<n;i++){
                caracter=getc(file);
                fputc(caracter,file_dest);
                }

    }
```

Seal of approval



Moreno Zarate Victor Gibrán

M. en C. Nidia Asunción Cortez Duarte
ESCOM - IPN
REVISADO

Pd ok 12/08/16
a,b ok

M. en C. Nidia Asunción Cortez Duarte
ESCOM - IPN
REVISADO
Cryptography

Shif. text ok
31/08/16 Imag OK

Affine V1 ok
07/09/16

Vigenere ok 07/09/16