Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Experiment 1

Date of Performance : 21-3-2022 **Date of Submission: 21-3-2022**

SAP Id: 60004190016 **Name :Aunali Patel**

Div: A **Batch : A3**

Aim of Experiment

Design and Implement Encryption and Decryption Algorithm for

a. Caesar cipher cryptographic algorithm by considering letter [A..Z] and digits [0..9]. Create two functions Encrypt() and Decrypt(). Apply Brute Force Attack to reveal secret. Create Function BruteForce(). Demonstrate the use of these functions on any paragraph.
b. Affine Cipher. Your Program Must Input Image in Gray Scale. Choose keys according to Gray Scale Intensity level. Create two functions Encrypt() and Decrypt(). Make sure to have Multiplicative Inverse Exists for one of the Key in selected Key pair of Affine Cipher. (CO1)

Theory / Algorithm / Conceptual Description

Caesar cipher: The Caesar cipher (or Caesar code) is a monoalphabetic substitution cipher, where each letter is replaced by another letter located a little further in the alphabet (therefore shifted but always the same for given cipher message). The shift distance is chosen by a number called the offset, which can be right (a to b) or left (b to a).

Affine Cipher: The Affine Cipher is a type of monoalphabetic substitution cipher, where each letter in an alphabet is mapped to its numeric equivalent, encrypted using a simple mathematical function, and converted back to a letter.

Program

A)

```
int encrypt(){
  cout << "enter plaintext: " <<endl;
  string s2,s = "" ;
  getline(cin, s);
  string m = "abcdefghijklmnopqrstuvwxyz0123456789" ;
  int key = 19;
  cout << "enter key: "<<endl;cin>>key;
  int n = key;
```

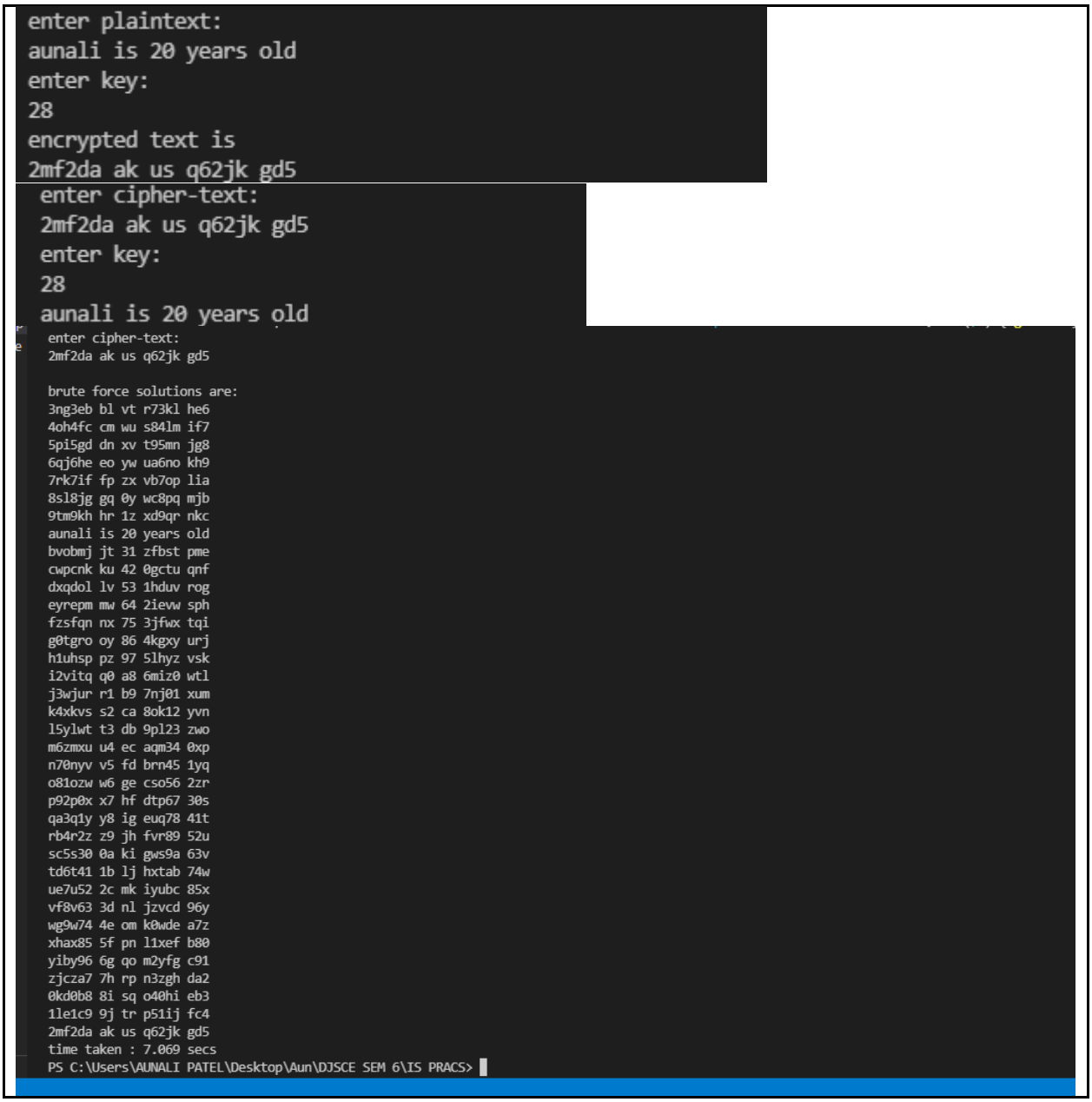
```
    for(int i = 0 ; i < 1 ; i++){
        string s1 = "";
        for(int j = 0 ; j < s.length() ; j++){
            if(s[j] == ' ' ){
                s1+= ' ';
                continue;
            }{
                int k = 0;
                while(s[j] != m[k]){
                                    k++;
                }
                s1 += m[(k+key)%36];
            }
        }
        cout << "encrypted text is "<<endl<<s1<<endl;
    }
    return 0;



int decrypt_with_key(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout << "enter cipher-text: " <<endl;
    string s2,s = "" ;
    getline(cin, s);
    string m = "abcdefghijklmnopqrstuvwxyz0123456789" ;
    int key = 1;cout<<"enter key: "<<endl;cin>>key;
    int n = -1*key + 36;
    for(int i = 0 ; i < 1 ; i++){
        string s1 = "";
        for(int j = 0 ; j < s.length() ; j++){
            if(s[j] == ' ' ){
                s1+= ' ';
                continue;
            }{
                int k = 0;
                while(s[j] != m[k]){
                    k++;
                }
                s1 += m[(k+n)%36];
            }
        }
        cout << s1<<endl;
    }
    return 0;
}
```

```
int bruteforce_attack(){
   cout << "enter cipher-text: " <<endl;
   string s2,s = "" ;
   getline(cin, s);
   string m = "abcdefghijklmnopqrstuvwxyz0123456789" ;
   int n = 0;
   cout << endl<<"brute force solutions are: "<<endl;
   for(int i = 0 ; i < 36 ; i++){
      n++;
      string s1 = "";
      for(int j = 0 ; j < s.length() ; j++){
         if(s[j] == ' ' ){
            s1+= ' ';
            continue;
         }{
            int k = 0;
            while(s[j] != m[k]){
               k++;
            }
            s1 += m[(k+n)%36];
         }
      }
      cout << s1<<endl;
   }
   return 0;
}
```

Output



Program

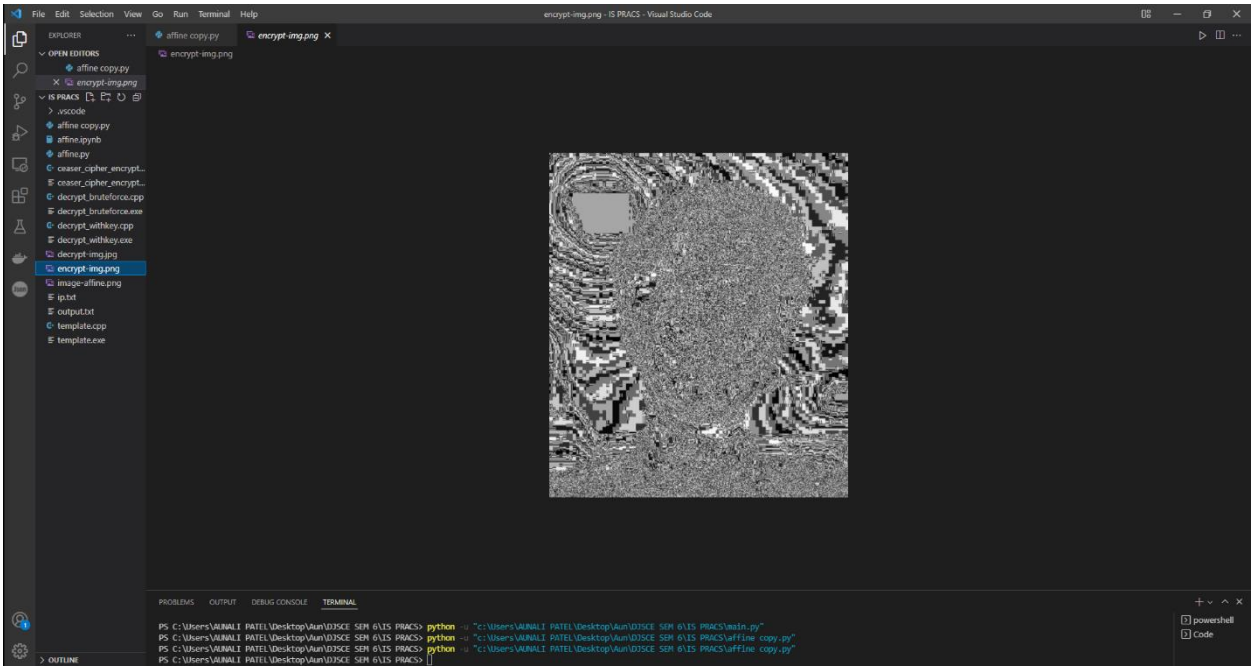
B)

Input

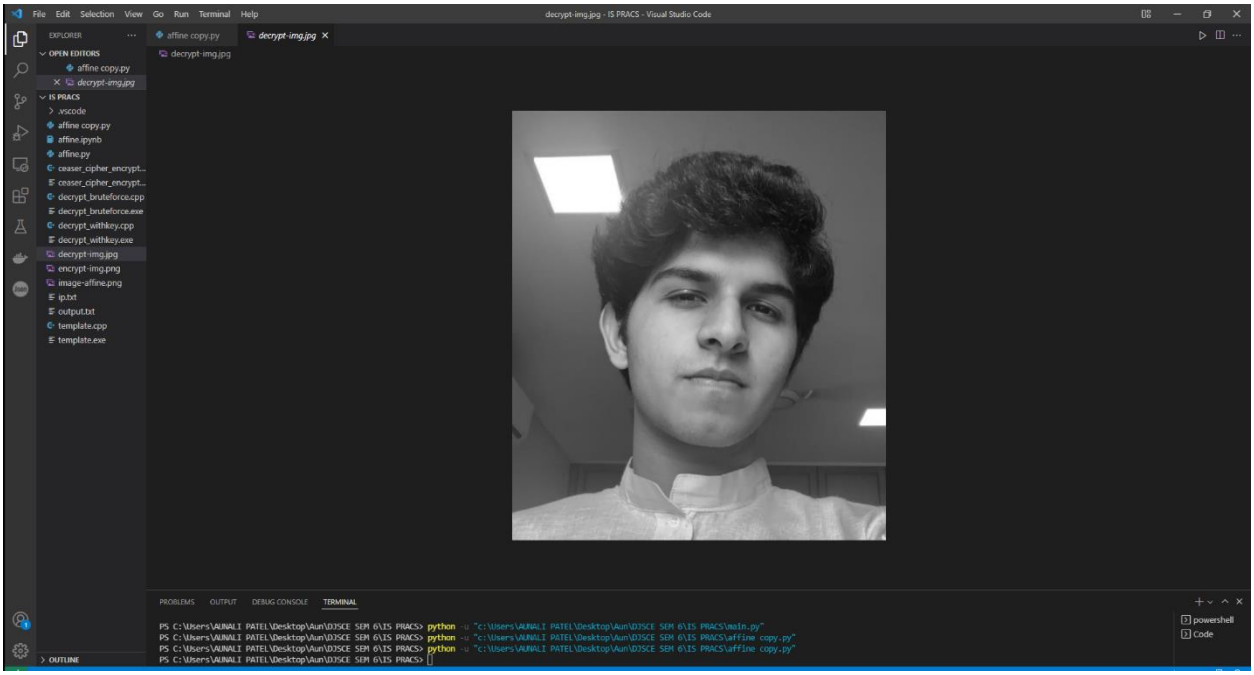
```
import cv2
import numpy as np
def decryption(img_name: str, key1, key2):
    img = cv2.imread(img_name, cv2.IMREAD_COLOR)
    m_inverse = pow(key1, -1, 256)
    decrypt_temp = (img - key2) % 256
    decrypt = (decrypt_temp * m_inverse) % 256
    cv2.imwrite('decrypt-img.jpg',decrypt)
def encryption(img_name: str, key1, key2):
    img = cv2.imread(img_name, cv2.IMREAD_COLOR)
    grayimg = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    grey_arr = np.asarray(grayimg)
    temp = (grey_arr * key1) % 256
    cipher = (temp + key2) % 256
    cv2.imwrite('encrypt-img.png',cipher)
key1 = 157
key2 = 17987
image_name = 'image-affine.png'
encryption(img_name=image_name, key1=key1, key2=key2)
decryption(img_name='encrypt-img.png', key1=key1, key2=key2)
```

Output

Encrypted image:



Decrypted image:



Actual image: