Objective 2

a. Write a program to implement Caesar Cipher encryption and decryption.

Caesar Cipher:

Procedure: Caesar Cipher is an Additive cipher which uses same key for both encryption and decryption. Process can be illustrated as follow.

Encryption: CT = (PT + KEY) mod 26

Decryption: PT = (CT – KEY) mod 26

Code:

```
import java.util.Scanner;
class CaesarCipher {
   public static final String lower = "abcdefghijklmnopqrstuvwxyz";
   public static final String upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

   static String encryption(String plainTxt, int key){
      String encrTxt="";
      for(int i = 0; i < plainTxt.length(); i++){
         char ch = plainTxt.charAt(i);
         if(plainTxt.charAt(i) >= 'a' && plainTxt.charAt(i) <= 'z'){
            int charPos = lower.indexOf(ch);
            int newPos = (charPos+key) % 26;
            encrTxt += (char)(lower.charAt(newPos));
         }
         else if(plainTxt.charAt(i) >= 'A' && plainTxt.charAt(i) <= 'Z'){
             int charPos = upper.indexOf(ch);
            int newPos = (charPos+key) % 26;
            encrTxt += (char)(upper.charAt(newPos));
         }
         else
            encrTxt += plainTxt.charAt(i);
      }
      return encrTxt;
   }

    static String decryption(String encrTxt, int key){
      String decrTxt="";
      for(int i = 0; i < encrTxt.length(); i++){
         char ch = encrTxt.charAt(i);
         if(ch >= 'a' && ch <= 'z'){
            int charPos = lower.indexOf(ch);
            int newPos = (charPos-key) % 26;
```

```
            if(newPos < 0)
               newPos = newPos + 26;
            decrTxt += (char)(lower.charAt(newPos));
         }
         else if(ch >= 'A' && ch <= 'Z'){
            int charPos = upper.indexOf(ch);
            int newPos = (charPos-key) % 26;
            if(newPos < 0)
               newPos = newPos + 26;
            decrTxt += (char)(upper.charAt(newPos));
         }
         else
            decrTxt += encrTxt.charAt(i);
      }
      return decrTxt;
   }

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String plainTxt;
      int key;
      System.out.print("\nEnter plain text: ");
      plainTxt = sc.nextLine();
      System.out.print("\nEnter key: ");
      key = sc.nextInt();
     String cipher = encryption(plainTxt, key);
     System.out.println("\nCipher Text: "+cipher);
     System.out.println("\nPlaint Text: "+ decryption(cipher, key));
   }
}
```

Output:

```
PS D:\DATAs\NITJ\CryptoLab> javac CaesarCipher.java
PS D:\DATAs\NITJ\CryptoLab> java CaesarCipher

Enter plain text: This text will be Encrypted and Decrypted.

Enter key: 21

Cipher Text: Ocdn ozso rdgg wz Zixmtkozy viy Yzxmtkozy.

Plaint Text: This text will be Encrypted and Decrypted.
PS D:\DATAs\NITJ\CryptoLab>
```

Objective 2

b. Write a program to implement Affine Cipher encryption and decryption.

Affine Cipher:

Procedure:

First, we take first key k1 which is coprime w.r.t. 26.
Second key will be any random number between 1 to 26.
Now we will apply Affine algorithm on each character of plain text as follow
Encryption:
CT = (PT * K1 + K2) mod 26
or
IT = (PT * K1) mod 26
CT = (IT + K2) mod 26

Decryption:
PT = (CT – K2) * $K1^{-1}$ mod 26
or
IT = (CT – K2) mod 26
PT = (IT * $K1^{-1}$) mod 26

Where, $K1^{-1}$ is a multiplicative inverse of K1.

Code:

```
import java.util.Scanner;
class AffineCipher{
static int k1_inverse(int n){
  int a, b, q, r, t1, t2, t;
  a = 26;
  b = n;
  t1 = 0;
  t2 = 1;
  System.out.println("q  a  b  r  t1  t2  t");
  while( b != 0){
    q = a/b;
    r = a%b;
    t = t1 - q*t2;
    System.out.println(q+"  "+ a +"  "+ b+"  "+ r +"  "+ t1 +"  "+ t2 +"  "+ t);
    a = b;
    b = r;
    t1 = t2;
    t2 = t;
  }
  if(t1 < 0)
```

```
    return t1+26;
  return t1;
}

static String encryptPt(char[] pt, int k1, int k2){
String cipher = "";
for(int i = 0; i < pt.length; i++)
{
  if(pt[i] != ' '){
    cipher = cipher + (char)(((k1 * (pt[i] - 'A') + k2)%26) + 'A');
  }
  else
    cipher += pt[i];

}
return cipher;
}

static String decryptCt(char[] cText, int k1_inv, int k2){
String pt = "";
for(int i = 0 ;i < cText.length; i++){
  if(cText[i] != ' '){
    pt = pt + (char)(((k1_inv * ((cText[i] + 'A' - k2)) % 26)) + 'A');
  }
  else
    pt += cText[i];
}
return pt;
}

static boolean checkCoPrime(int n){
  if(n >0 || n < 26){
    if(n%2!=0 && 26%n !=0)
          return true;
  }
  return false;
}

public static void main(String args[]){
  Scanner sc = new Scanner(System.in);
  String pt;
  String ct;
```

```
  String decTxt;
  int k1;
  int k2;
int k1_inv = 0;
  System.out.print("Enter plain text: ");
  pt = sc.nextLine();
  System.out.print("\n Enter key k1: ");
  k1 = sc.nextInt();
  while(!checkCoPrime(k1)){
    System.out.print("\n Invalid key, enter valid key: ");
    k1 = sc.nextInt();
  }
  System.out.print("\n Enter key k2: ");
  k2 = sc.nextInt();
  k1_inv = k1_inverse(k1);
  System.out.println("Inverse of "+k1+" = "+ k1_inv);
  ct = encryptPt(pt.toCharArray(), k1, k2);
  System.out.println("Cipher Text: " + ct);
  decTxt = decryptCt(ct.toCharArray(), k1_inv, k2);
 System.out.println("Plaint Text: "+ decTxt);
}
}
```

Output:

```
Enter plain text: DR BR NIT JALANDHAR

Enter key k1: 15

Enter key k2: 22

q  a  b  r  t1  t2  t
1  26  15  11  0  1  -1
1  15  11  4  1  -1  2
2  11  4  3  -1  2  -5
1  4  3  1  2  -5  7
3  3  1  0  -5  7  -26

Inverse of 15 = 7

Cipher Text: PR LR JMV BWFWJPXWR

Plaint Text: DR BR NIT JALANDHAR
```