1 (a) Caesar Cipher *(Encryption + Decryption)*

Aim

To write a Java program that performs encryption and decryption using the Caesar-cipher algorithm.

Algorithm (shift = 3)

1. Read the plaintext string.
2. **Encryption:** for each letter *P*, compute $C = (P + 3) \bmod 26$.
3. **Decryption:** for each letter *C*, compute $P = (C - 3) \bmod 26$ (or equivalently $(C + 23) \bmod 26$).
4. Display both cipher text and recovered plain text.

Program (Java, minimal)

```
public class CaesarCipher {
    private static final String ABC = "abcdefghijklmnopqrstuvwxyz";
    private static String xform(String t,int s){
        StringBuilder sb=new StringBuilder();
        for(char ch:t.toLowerCase().toCharArray()){
            int i=ABC.indexOf(ch);
            sb.append(i==-1?ch:ABC.charAt((i+s+26)%26));
        }
        return sb.toString();
    }
    public static String encrypt(String p){ return xform(p,3); }
    public static String decrypt(String c){ return xform(c,-3); }
    public static void main(String[] a){
        String p="computer";
        String c=encrypt(p);
        System.out.println(c);           // frpsxwhu
        System.out.println(decrypt(c));  // computer
    }
}
```

Sample Output

```
frpsxwhu
computer
```

(The values match the handbook example.)

Result

Thus, a Java program to perform encryption and decryption using the Caesar-cipher algorithm was executed successfully.

---

1 (b) Playfair Cipher *(Encryption + Decryption)*

Aim

To write a Java program that performs encryption and decryption using the Playfair-cipher technique.

Algorithm (brief)

1. Build a 5 × 5 matrix from the keyword (I/J share one cell).
2. Pre-process plaintext in digraphs, inserting **x** between repeated letters and padding odd length.
3. For each pair, apply Playfair rules:
    - Same row → take the letter to the right (wrap-around).
    - Same column → take the letter below (wrap-around).
    - Rectangle → substitute each letter with the one in its row and the other's column.
4. For decryption, reverse the shifts (left / up).

Program (Java, compact)

```
import java.util.*;
public class PlayfairCipher{
    private final char[][] box=new char[5][5];
    private final Map<Character,int[]> loc=new HashMap<>();
    public PlayfairCipher(String key){
        key=
(key+"abcdefghiklmnopqrstuvwxyz").toLowerCase().replace("j","i").replaceAll("[^a-
z]","");
        int k=0; for(char c:key.toCharArray())
            if(!loc.containsKey(c)){ box[k/5][k%5]=c; loc.put(c,new int[]
{k/5,k%5}); k++; }
    }
    public String enc(String msg){ return proc(msg,false); }
    public String dec(String msg){ return proc(msg,true);  }
    private String proc(String text,boolean dec){
        text=text.toLowerCase().replace("j","i").replaceAll("[^a-z]","");
        StringBuilder sb=new StringBuilder();
        for(int i=0;i<text.length();i+=2){
            char a=text.charAt(i), b=(i+1<text.length()&&text.charAt(i+1)!=a)?
text.charAt(i+1):'x';
            int[] p1=loc.get(a), p2=loc.get(b);
            int r1=p1[0],c1=p1[1], r2=p2[0],c2=p2[1];
            if(r1==r2){ c1=(c1+(dec?-1:1)+5)%5; c2=(c2+(dec?-1:1)+5)%5; }
            else if(c1==c2){ r1=(r1+(dec?-1:1)+5)%5; r2=(r2+(dec?-1:1)+5)%5; }
            else{ int tmp=c1; c1=c2; c2=tmp; }
            sb.append(box[r1][c1]).append(box[r2][c2]);
        }
        return sb.toString();
    }
    public static void main(String[] a){
        PlayfairCipher pf=new PlayfairCipher("security");
```

```
        String c=pf.enc("cryptography");
        System.out.println(c);            // usbnamkcboga
        System.out.println(pf.dec(c));    // cryptography
    }
}
```

Sample Output

```
usbnamkcboga
cryptography
```

Matches the manual's trace.

Result

Thus, a Java program to perform encryption and decryption using the Playfair-cipher algorithm was executed successfully.

---

1 (c) Hill Cipher *(Encryption only)*

Aim

To write a Java program to perform encryption using the Hill-cipher algorithm.

Algorithm (core)

1. Group plaintext in blocks of m letters (here $m = 3$). Pad with **x** if needed.
2. Convert each block to a vector **P** of numbers *(a = 0 … z = 25)*.
3. Multiply **C = P · K (mod 26)** using an invertible 3 × 3 key matrix **K**.
4. Map the numeric result back to letters to obtain cipher text.

Program (Java, minimal)

```
public class HillCipher{
    private static final int[][] K={{17,17,5},{21,18,21},{2,2,19}}; // invertible
    private static int mod(int x){ return (x%26+26)%26; }
    public static String encrypt(String plain){
        plain=plain.toLowerCase().replaceAll("[^a-z]","");
        while(plain.length()%3!=0) plain+='x';
        StringBuilder out=new StringBuilder();
        for(int i=0;i<plain.length();i+=3){
            int[] p=
{plain.charAt(i)-97,plain.charAt(i+1)-97,plain.charAt(i+2)-97};
            for(int r=0;r<3;r++){
                int sum=K[r][0]*p[0]+K[r][1]*p[1]+K[r][2]*p[2];
                out.append((char)(mod(sum)+97));
            }
        }
```

```
        return out.toString();
    }
    public static void main(String[] a){
        System.out.println(encrypt("paymoremoney")); // lnshdlewmtrw
    }
}
```

Sample Output

```
lnshdlewmtrw
```

(Identical to the handbook.)

Result

Thus, a Java program to perform encryption using the Hill-cipher algorithm was executed successfully.

---

1 (d) Vigenère Cipher *(Encryption + Decryption)*

Aim

To write a Java program that performs encryption and decryption using the Vigenère-cipher technique.

Algorithm (essentials)

1. Read plaintext and repeated-keyword key.
2. **Encryption:** for each position *i*, compute
   $C_i = (P_i + K_i) \bmod 26$.
3. **Decryption:** compute
   $P_i = (C_i - K_i) \bmod 26$.
4. Display cipher text and recovered plain text.

Program (Java, concise)

```
public class VigenereCipher{
    private static String ks(String k,int n){ StringBuilder s=new StringBuilder();
for(int i=0;i<n;i++) s.append(k.charAt(i%k.length())); return s.toString(); }
    private static String proc(String t,String k,boolean dec){
        t=t.toLowerCase().replaceAll("[^a-z]",""); k=k.toLowerCase();
        StringBuilder sb=new StringBuilder(); String ks=ks(k,t.length());
        for(int i=0;i<t.length();i++){
            int p=t.charAt(i)-97, ki=ks.charAt(i)-97;
            sb.append((char)(((dec?p-ki:p+ki)+26)%26+97));
        }
        return sb.toString();
    }
    public static String encrypt(String p,String k){ return proc(p,k,false); }
    public static String decrypt(String c,String k){ return proc(c,k,true ); }
```

```
    public static void main(String[] a){
        String c=encrypt("wearediscoveredsaveyourself","deceptive");
        System.out.println(c);                          //
zicvtwqngrzgvtwavzhcqyglmgj
        System.out.println(decrypt(c,"deceptive"));  //
wearediscoveredsaveyourself
    }
}
```

Sample Output

```
zicvtwqngrzgvtwavzhcqyglmgj
wearediscoveredsaveyourself
```

(As shown in the lab hand-out.)

Result

Thus, a Java program to perform encryption and decryption using the Vigenère-cipher algorithm was executed successfully.