Government College of Engineering, Jalgaon
(An Autonomous Institute of Govt. of Maharashtra)

Name: **PRN:**

Class: L.Y **Semester:** VII **Batch:**

Date of Performance: _____ **Date of Completion:** _____

Subject: CO407U CNSL **Subject Teacher:** Ms. Shrutika Mahajan

Practical – 4

Aim: Implementation of any two algorithms

a. Ceaser cipher

b. Hill Cipher

c. Substitution Cipher

Theory:

1. Caesar Cipher

Introduction:

The Caesar Cipher is one of the earliest and simplest encryption techniques. It is a type of substitution cipher, in which each letter in the plaintext is shifted by a fixed number of positions down the alphabet.

Working Principle:

- Each character in the message is replaced with another character obtained by shifting it by a key value.
- For example, with a shift of 3:
  - A → D, B → E, C → F, …, X → A, Y → B, Z → C.
- The process is cyclic, meaning after Z, it goes back to A.

Encryption:

Formula for encryption is:

C = (P + K) mod 26

Where,

- C = Ciphertext letter
- P = Plaintext letter (0–25)
- K = Key (number of shifts)

Decryption:

Formula for decryption is:

P = (C - K) mod 26

Where the terms have the same meaning as above.

Features:

- Simple to implement.
- Vulnerable to brute force (only 25 possible keys).
- Used mainly for learning cryptography basics.

Code:

```
package string;
import java.util.Scanner;
public class CaesarCipherSimple {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter the text: ");
      String text = sc.nextLine();
      System.out.print("Enter the shift value: ");
      int shift = sc.nextInt();
      String encrypted = "";
      String decrypted = "";
      for (char c : text.toCharArray()) {
         if (Character.isLetter(c)) {
            char base = Character.isUpperCase(c) ? 'A' : 'a';
            encrypted += (char) ((c - base + shift) % 26 + base);
         } else {
            encrypted += c;
         }
      }
      for (char c : encrypted.toCharArray()) {
```

```
            if (Character.isLetter(c)) {
                char base = Character.isUpperCase(c) ? 'A' : 'a';
                decrypted += (char) ((c - base - shift + 26) % 26 + base);
            } else {
                decrypted += c;
            }
        }
        System.out.println("\nOriginal: " + text);
        System.out.println("Encrypted: " + encrypted);
        System.out.println("Decrypted: " + decrypted);
        sc.close();
    }
}
```

Output:

```
Enter the text: HELLO WORLD
Enter the shift value: 2

Original: HELLO WORLD
Encrypted: JGNNQ YQTNF
Decrypted: HELLO WORLD
```

2. Hill Cipher

Introduction:

The Hill Cipher is a polygraphic substitution cipher invented by Lester S. Hill in 1929. It is based on linear algebra and makes use of matrix multiplication to encrypt and decrypt messages.

Working Principle:

- It considers blocks of letters (instead of single letters like Caesar).
- Each block is represented as a vector of numbers (A = 0, B = 1, …, Z = 25).
- A **key matrix** is chosen, and matrix multiplication (mod 26) is performed.

Encryption:

If P is the plaintext vector, K is the key matrix, then the ciphertext C is:

$C = K \times P \pmod{26}$

Decryption:

To decrypt, the inverse of the key matrix K^{-1} (mod 26) is used:

$P = K^{-1} \times C \pmod{26}$ **Features:**

- Stronger than Caesar Cipher because it encrypts multiple letters at once.
- Depends on the invertibility of the key matrix (must have a non-zero determinant and gcd(det, 26) = 1).
- Provides diffusion: a change in one letter of plaintext affects multiple letters of ciphertext.

Code:

```
package string;
import java.util.*;
public class HillCipherFull {
    static String encrypt(String msg, int[][] key) {
        msg = msg.toUpperCase().replaceAll("[^A-Z]", "");
        if (msg.length() % 2 != 0) msg += "X";
        StringBuilder res = new StringBuilder();
        for (int i = 0; i < msg.length(); i += 2) {
            int p1 = msg.charAt(i) - 'A';
            int p2 = msg.charAt(i + 1) - 'A';
            int c1 = (key[0][0]*p1 + key[0][1]*p2) % 26;
            int c2 = (key[1][0]*p1 + key[1][1]*p2) % 26;
            res.append((char)(c1 + 'A')).append((char)(c2 + 'A'));
        }
        return res.toString();
    }
    static int modInverse(int a, int m) {
        a = a % m;
        for (int x = 1; x < m; x++)
            if ((a * x) % m == 1) return x;
        return -1;
    }
    static int[][] inverseKey(int[][] key) {
        int det = (key[0][0]*key[1][1] - key[0][1]*key[1][0]) % 26;
        if (det < 0) det += 26;
        int detInv = modInverse(det, 26);
        if (detInv == -1) throw new IllegalArgumentException("Key not invertible!");
        int[][] inv = new int[2][2];
        inv[0][0] = ( key[1][1]*detInv ) % 26;
```

```
        inv[0][1] = (-key[0][1]*detInv + 26) % 26;
        inv[1][0] = (-key[1][0]*detInv + 26) % 26;
        inv[1][1] = ( key[0][0]*detInv ) % 26;
        return inv;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[][] key = new int[2][2];
        System.out.println("Enter 2x2 key matrix (numbers 0-25, invertible mod 26):");
        for (int i = 0; i < 2; i++)
            for (int j = 0; j < 2; j++)
                key[i][j] = sc.nextInt();
        sc.nextLine();
        System.out.print("Enter text to encrypt: ");
        String text = sc.nextLine();
        String encrypted = encrypt(text, key);
        System.out.println("Encrypted Text: " + encrypted);
        int[][] invKey = inverseKey(key);
        String decrypted = encrypt(encrypted, invKey);
        System.out.println("Decrypted Text: " + decrypted);
        sc.close();
    }
}
```

Output:

```
Enter 2x2 key matrix (numbers 0-25, invertible mod 26):
3 3
2 5
Enter text to encrypt: POH
Encrypted Text: JWMZ
Decrypted Text: POHX
```

3. Substitution Cipher

Definition:
A substitution cipher is a method of encryption where each element of the plaintext (like a letter, number, or symbol) is replaced by another element according to a fixed system or key. The positions of the characters are not changed, only the characters themselves are substituted.

Types of Substitution Ciphers:

1. **Simple (Monoalphabetic) Substitution Cipher:**
    - Each letter of the plaintext is replaced by a unique letter of the ciphertext.
    - Example: If the key shifts A → D, B → E, C → F, then "ABC" becomes "DEF". ○ **Limitation:** Vulnerable to frequency analysis because each letter has a consistent mapping.
2. **Polyalphabetic Substitution Cipher:**
    - Uses multiple substitution alphabets to encrypt the plaintext.
    - Example: **Vigenère Cipher**, where a key word is repeated to shift letters differently across the message.
    - **Advantage:** More secure than simple substitution, harder to break with frequency analysis.
3. **Homophonic Substitution Cipher:**
    - Each plaintext letter can be replaced by more than one ciphertext symbol. ○ Reduces patterns in frequency, making cryptanalysis more difficult.

Key Concept:

- The **key** is the system used for substitution. Without the key, decrypting the ciphertext is difficult.

Encryption Process:

1. Choose a key (substitution rule).
2. Replace each plaintext character according to the key.
3. The output is the ciphertext.

Decryption Process:

1. Use the same key (or inverse key in symmetric systems).
2. Replace each ciphertext character with the corresponding plaintext character.
3. Recover the original message.

Advantages:

- Easy to implement.
- Can be combined with other ciphers for stronger security.

Disadvantages:

- Monoalphabetic substitution is easy to break using frequency analysis.
- Requires secure key management.

Applications:

- Historical encryption (Caesar cipher, simple substitution ciphers).

- Modern cryptography mostly uses more complex algorithms, but substitution principles are foundational.

Code:

```
package string;

import java.util.Scanner;
public class SubstitutionCipher {
    static void printMapping(String key) {
      System.out.println("Substitution Mapping:");
      System.out.println("Plain : ABCDEFGHIJKLMNOPQRSTUVWXYZ");
      System.out.println("Cipher: " + key);
      System.out.println();
   }
   static String encrypt(String message, String key) {
    message = message.toUpperCase().replaceAll("[^A-Z]", "");
    StringBuilder encrypted = new StringBuilder();
    for (char c : message.toCharArray()) {
            encrypted.append(key.charAt(c - 'A'));
      }
      return encrypted.toString();
   }
   static String decrypt(String cipher, String key) {
      StringBuilder decrypted = new StringBuilder();

      for (char c : cipher.toCharArray()) {
            int index = key.indexOf(c);
            decrypted.append((char) (index + 'A'));
      }
      return decrypted.toString();
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String key = "QWERTYUIOPASDFGHJKLZXCVBNM";
      printMapping(key);
      System.out.print("Enter message: ");
      String message = sc.nextLine();
      String encrypted = encrypt(message, key);
      System.out.println("Encrypted: " + encrypted);
      String decrypted = decrypt(encrypted, key);
      System.out.println("Decrypted: " + decrypted);
      sc.close();
   }
```

}

Output:

```
Substitution Mapping:
Plain : ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher: QWERTYUIOPASDFGHJKLZXCVBNM

Enter message: SAD
Encrypted: LQR
Decrypted: SAD
```

Conclusion:

1. **Caesar Cipher:**
   - A simple monoalphabetic substitution cipher that shifts letters by a fixed number.
   - Easy to implement but insecure for long messages due to brute-force vulnerability.
   - Useful to understand basic encryption and decryption concepts.
2. **Hill Cipher:**
   - A polygraphic substitution cipher using a key matrix and matrix multiplication. o More secure than simple substitution because it encrypts multiple letters at once.
   - Demonstrates the use of linear algebra in cryptography and enables decryption using an inverse key matrix.
3. **Substitution Cipher:**
   - Replaces each plaintext character with another character according to a fixed mapping.
   - Simple to implement; monoalphabetic version is vulnerable to frequency analysis, while polyalphabetic versions are more secure.
   - Helps understand the principle of character substitution, which is foundational for modern encryption methods.

Course Teacher
Ms. Shrutika Mahajan