Experiment No. 1

Aim: Design and Implementation of a product cipher using Substitution and Transposition

Program:

```
// package anproduct;
import java.io.*;

public class Anproduct {
    public static void main(String[] args) throws IOException {
        System.out.println("Enter choice");
        System.out.append("1.Encryption\n2.Decryption");
        BufferedReader obj = new BufferedReader(new InputStreamReader(System.in));
        int ch = Integer.parseInt(obj.readLine());
        if (ch == 1) {
            enc e = new enc();
            e.enc();
        } else if (ch == 2) {
            dec d = new dec();
            d.dec();
        } else {
            System.out.println("Invalid Choice");
        }
    }
}

class enc {
    public void enc() throws IOException {
        BufferedReader obj = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter IP:");
        String ip = obj.readLine();
        System.out.println("Enter Key 1:");
        int k1 = Integer.parseInt(obj.readLine());
        System.out.println("Enter Key 2:");
        int k2 = Integer.parseInt(obj.readLine());
        String op = "";
        op = additive(ip, k1);
        op = tansposition(op, k2);
        System.out.println(op);
    }

    String additive(String ip, int k) {
        StringBuilder s = new StringBuilder();
```

```
    int len = ip.length();
    char temp;
    int t1;
    for (int i = 0; i < len; i++) {
        temp = ip.charAt(i);
        if (Character.isUpperCase(temp)) {
            t1 = (int) temp - (int) 'A';
            t1 = (t1 + k) % 26;
            t1 = t1 + (int) 'A';
            temp = (char) t1;
            s.append(temp);
        } else if (Character.isLowerCase(temp)) {
            t1 = (int) temp - (int) 'a';
            t1 = (t1 + k) % 26;
            t1 = t1 + (int) 'a';
            temp = (char) t1;
            s.append(temp);
        } else {
            s.append(temp);
        }
    }
    String op = s.toString();
    return op;
}

String tansposition(String ip, int m_row) {
    char op[][] = new char[100][100];
    int len = ip.length();
    String op2 = "";
    int i1, i2, i;
    // calculate columns from rows
    int m_col = (int) Math.ceil((float) len / m_row);
    for (i = 0, i1 = 0, i2 = 0; i < len; i++) {
        op[i2][i1] = ip.charAt(i);
        i2++;
        if (i2 == m_row) {
            i2 = 0;
            i1++;
        }
    }
    for (i1 = 0; i1 < m_row; i1++) {
        for (i2 = 0; i2 < m_col; i2++) {
            op2 = op2 + op[i1][i2];
        }
```

```
        }
        return (op2);
    }
}

class dec {
    public void dec() throws IOException {
        BufferedReader obj = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter IP:");
        String ip = obj.readLine();
        System.out.println("Enter Key 1:");
        int k1 = Integer.parseInt(obj.readLine());
        System.out.println("Enter Key 2:");
        int k2 = Integer.parseInt(obj.readLine());
        String op = "";
        op = additive(ip, k1);
        op = tansposition(op, k2);
        System.out.println(op);
    }

    String additive(String ip, int k) {
        StringBuilder s = new StringBuilder();
        int len = ip.length();
        char temp;
        int t1;
        for (int i = 0; i < len; i++) {
            temp = ip.charAt(i);
            if (Character.isUpperCase(temp)) {
                t1 = (int) temp - (int) 'A';
                t1 = (t1 - k + 26) % 26;
                t1 = t1 + (int) 'A';
                temp = (char) t1;
                s.append(temp);
            } else if (Character.isLowerCase(temp)) {
                t1 = (int) temp - (int) 'a';
                t1 = (t1 - k + 26) % 26;
                t1 = t1 + (int) 'a';
                temp = (char) t1;
                s.append(temp);
            } else {
                s.append(temp);
            }
        }
        String op = s.toString();
```

```
        return op;
    }

    String tansposition(String ip, int m_row) {
        char op[][] = new char[100][100];
        int len = ip.length();
        String op2 = "";
        int i1, i2, i;
        // calculate columns from rows
        int m_col = (int) Math.ceil((float) len / m_row);
        for (i = 0, i1 = 0, i2 = 0; i < len; i++) {
            op[i1][i2] = ip.charAt(i);
            i2++;
            if (i2 == m_col) {
                i2 = 0;
                i1++;
            }
        }
        for (i1 = 0; i1 < m_col; i1++) {
            for (i2 = 0; i2 < m_row; i2++) {
                op2 = op2 + op[i2][i1];
            }
        }
        return (op2);
    }
}
```

Output:

```
Enter choice
1.Encryption
2.Decryption1
Enter IP:
CRYPTOGRAPHY
Enter Key 1:
1
Enter Key 2:
5
DPISHZZS QB UQ
```