LAB – 2

Aim: Write a program to implement

1. Extended Euclidean Algorithm for finding multiplicative inverse
2. Multiplicative Cipher
3. Affine Cipher

1) Extended Euclidean Algorithm :

```
import java.util.*;
import java.math.BigInteger;

public class Euclidean
{
   public static int getGcd(int a, int b) {
      BigInteger b1 = BigInteger.valueOf(a);
      BigInteger b2 = BigInteger.valueOf(b);
      BigInteger gcd =b1.gcd(b2);

      return gcd.intValue();
   }

   public static int euclideanAlgo(int a, int b) {
      int r0 = a, r1 = b, s0 = 1, s1 = 0, t0 = 0, t1 = 1;
      int r, s, t, q;

      while(r1 > 0) {
         q = r0 / r1;

         r = r0 - (q * r1);
         r0 = r1;
         r1 = r;

         s = s0 - (q * s1);
         s0 = s1;
         s1 = s;

         t = t0 - (q * t1);
         t0 = t1;
         t1 = t;
      }

      t = t0;

      return t;
```

```
    }

    public static void main(String[] args) {
        System.out.print(euclideanAlgo(75, 21));
    }
}
```

OUTPUT :

```
-7
```

2) Multiplicative Cipher :

```
public class MultiplicativeCipher {
    public static String encrypt(String pt, int key) {
        StringBuilder encrypt =  new StringBuilder();

        for(int i = 0; i < pt.length(); i++) {
            int ascii = ((pt.charAt(i) - 'a') * key) % 26;
            encrypt.append((char)(ascii + 'a'));
        }

        return encrypt.toString();
    }

    public static String decrypt(String pt, int key) {
        StringBuilder decrypt =  new StringBuilder();
        MultiplicativeInverse mi = new MultiplicativeInverse();
        int inverse = mi.multiplicativeInverse(26, key);

        for(int i = 0; i < pt.length(); i++) {
            int ascii = ((pt.charAt(i) - 'a') * inverse) % 26;
            decrypt.append((char)(ascii + 'a'));
        }

        return decrypt.toString();
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Plain-Text : ");
        String pt = sc.nextLine();
        System.out.print("Enter Key : ");
        int n = sc.nextInt();
        String encrypted = encrypt(pt, n);
        String decrypted = decrypt(encrypted, n);
```

```
        System.out.println("Encription : " + encrypted);
        System.out.println("Decription : " + decrypted);
    }
}
```

OUTPUT :

```
Enter Plain-Text : welcome
Enter Key : 5
Encription : gudksiu
Decription : welcome
```

3) Affine Cipher :

```
public class AffineCipher {
    public static String encrypt(String pt, int key1, int key2) {
        StringBuilder encrypt =  new StringBuilder();

        for(int i = 0; i < pt.length(); i++) {
            int ascii = ((pt.charAt(i) - 'a') * key1 + key2) % 26;
            encrypt.append((char)(ascii + 'a'));
        }

        return encrypt.toString();
    }

    public static String decrypt(String pt, int key1, int key2) {
        StringBuilder decrypt =  new StringBuilder();
        MultiplicativeInverse mi = new MultiplicativeInverse();
        int inverse = mi.multiplicativeInverse(26, key1);

        for(int i = 0; i < pt.length(); i++) {
            int ascii = (((pt.charAt(i) - 'a') - key2) * inverse) % 26;
            decrypt.append((char)(ascii + 'a'));
        }

        return decrypt.toString();
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Plain-Text : ");
        String pt = sc.nextLine();
```

```
        System.out.print("Enter Key1 : ");
        int key1 = sc.nextInt();
        System.out.print("Enter Key2 : ");
        int key2 = sc.nextInt();

        String encrypted = encrypt(pt, key1, key2);
        String decrypted = decrypt(encrypted, key1, key2);
        System.out.println("Encription : " + encrypted);
        System.out.println("Decription : " + decrypted);
    }
}
```

OUTPUT :

```
Enter Plain-Text : welcome
Enter Key1 : 5
Enter Key2 : 1
Encription : hveltjv
Decription : welcome
```