Practical no 1: Substitution Cipher

Aim: Write programs to implement the following Substitution Cipher Techniques: Caesar Cipher and Monoalphabetic Cipher

1. Caesar Cipher

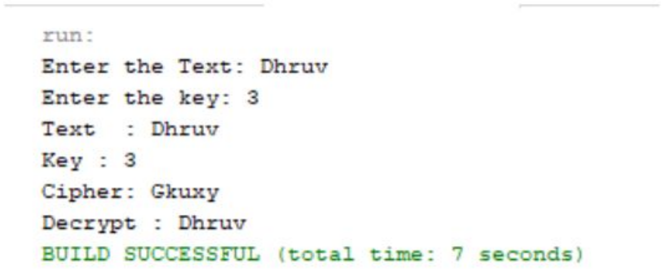
Code:

```
import java.util.*;
public class Caesar {
  // Encrypts text using a shift of s
  public static StringBuffer
encrypt(String text, int s)
  {  StringBuffer result= new
StringBuffer();
    for (int i=0; i<text.length(); i++)
    {
      if
(Character.isUpperCase(text.charAt(i)))
      {
        char ch =
(char)(((int)text.charAt(i) + s - 65) % 26 +
65);
        result.append(ch);
      }
      else
      {
        char ch =
(char)(((int)text.charAt(i) + s - 97) % 26 +
97);
        result.append(ch);
      }
    }
    return result;
  }
```

```
  public static StringBuffer
decrypt(StringBuffer text,int s)
  {
    StringBuffer result= new
StringBuffer();
    for (int i=0; i<text.length(); i++)
    {
      if
(Character.isUpperCase(text.charAt(i)))
      {
        char ch =
(char)(((int)text.charAt(i) - s - 65) % 26 +
65);
        result.append(ch);
      }
      else
      {
        int
midresult=((int)text.charAt(i) - s - 97);
        if (midresult < 0){
          midresult=midresult+26;
        }
        else{
          midresult=midresult;
        }
        char ch = (char)(midresult %
26 + 97);
        result.append(ch);
```

```
         }
      }
      return result;
   }
   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 key: ");
      int s = sc.nextInt();
      System.out.println("Text  : " + text);
      System.out.println("Key : " + s);
      System.out.println("Cipher: " +
encrypt(text, s));
      StringBuffer result = new
StringBuffer();
      result=encrypt(text, s);
      System.out.println("Decrypt : " +
decrypt(result, s));
   }
}
```

Output:



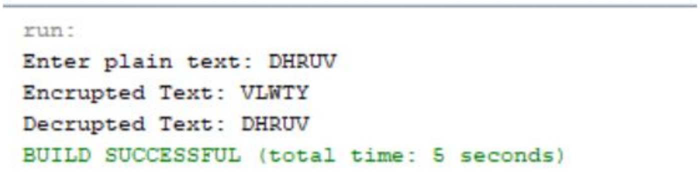
```
run:
Enter the Text: Dhruv
Enter the key: 3
Text  : Dhruv
Key : 3
Cipher: Gkuxy
Decrypt : Dhruv
BUILD SUCCESSFUL (total time: 7 seconds)
```

2. Monoalphabetic Cipher

Code:

```
import java.util.*;
public class Monoalphabetic{
   public static String
encrupt_text(String text,String key){
      StringBuffer buffer=new
StringBuffer(text);
      for(int i=0;i<buffer.length();i++){
         int index;
         char cipher_letter;
         index=buffer.charAt(i)-65;
         cipher_letter=key.charAt(index);
         buffer.setCharAt(i,cipher_letter);
      }
      String encrupted_text=new
String(buffer);
      return encrupted_text;
   }

   public static String
decrupt_text(String text,String key){
        StringBuffer buffer=new
StringBuffer(text);
        for(int i=0;i<buffer.length();i++){
            int index;
            char plain_letter;
            char letter=buffer.charAt(i);
            index=key.indexOf(letter);
            plain_letter=(char)(65+index);
            buffer.setCharAt(i,plain_letter);
        }
        String decrupted_text=new
String(buffer);
        return decrupted_text;
    }

    public static void main(String[] args)
{
        String
key="ZXCVBNMLKJHGFDSAQWERTYUIOP"
;
        Scanner sc=new
Scanner(System.in);
        System.out.print("Enter plain text:
");
        String pt=sc.next().toUpperCase();
        String cipher=encrupt_text(pt,key);
        String
decrupted_text=decrupt_text(cipher,ke
y);
        System.out.println("Encrypted
Text: " + cipher);
        System.out.println("Decrypted
Text: "+ decrupted_text);
    }
}
```

Output:



```
run:
Enter plain text: DHRUV
Encrupted Text: VLWTY
Decrupted Text: DHRUV
BUILD SUCCESSFUL (total time: 5 seconds)
```

Practical no 3: TranspositionCipher

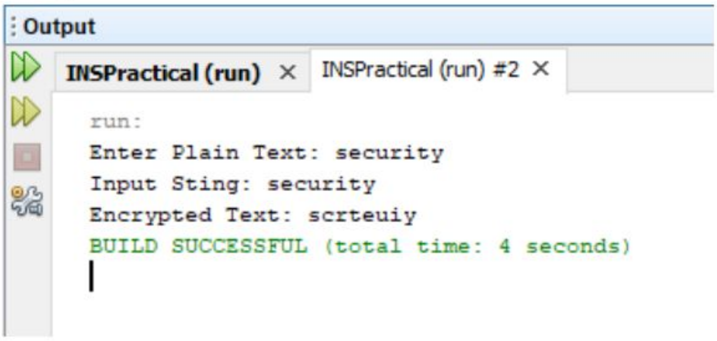
Aim: Write programs to implement the following Transposition Cipher Techniques: Rail Fence Cipher and Simple Columnar Technique

1. Rail Fence Cipher

Code:

```
import java.io.*;
import java.util.*;
public class RailFence {


   public static void encrypt(String
text,int fence){
       text=text.replaceAll(" ","");
       String output1=" ";
       String output2=" ";
       int len=text.length();
       System.out.println("Input Sting:
"+text);

       for (int i=0;i<len;i++){
          if(i%fence==0){

output1=output1+text.charAt(i);
          }
          else{
output2=output2+text.charAt(i);
          }
       }
       String output=output1+output2;
       output=output.replaceAll(" ","");
       System.out.println("Encrypted
Text: " + output);
    }

    public static void main(String[] args)
throws IOException {

       Scanner sc=new
Scanner(System.in);
      System.out.print("Enter Plain Text:
");
       String input=sc.next();
       encrypt(input,2);
    }
}
```

Output:



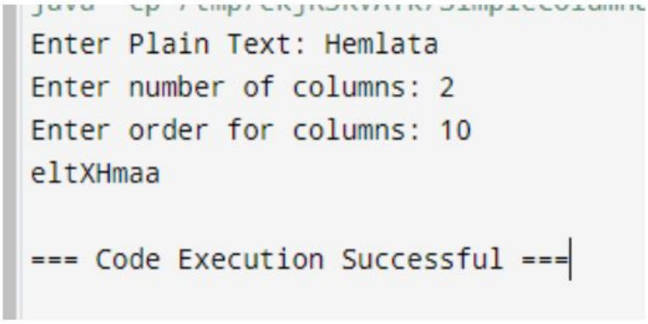
```
run:
Enter Plain Text: security
Input Sting: security
Encrypted Text: scrteuiy
BUILD SUCCESSFUL (total time: 4 seconds)
```

2. Simple Columnar Technique

Code:

```
import java.util.*;
public class SimpleColumnar {
   public static void main(String[] args){

      StringBuffer txt=new
StringBuffer();
      Scanner sc=new
Scanner(System.in);
      System.out.print("Enter Plain Text:
");
      String pt=sc.next();
      System.out.print("Enter number of
columns: ");
      int cols=sc.nextInt();
      System.out.print("Enter order for
columns: ");
      String order=sc.next();
      int rows = (int) Math.ceil((double)
pt.length() / cols);

      char[][] matrix=new
char[rows][cols];
      int n=0;
      for(int i=0;i<rows;i++){
         for(int j=0;j<cols;j++){
            if(n<pt.length()){
               matrix[i][j]=pt.charAt(n);
               n++;
            }
            else{
               matrix[i][j]='X';
            }
         }
      }
      int col_index=0;
      for(int i=0;i<order.length();i++){
         char new_option =
order.charAt(i);
         col_index = new_option - '0';
         for(int j=0;j<rows;j++){

txt.append(matrix[j][col_index]);
            }
      }
      System.out.println(txt.toString());
      }
}
```

Output:



```
Enter Plain Text: Hemlata
Enter number of columns: 2
Enter order for columns: 10
eltXHmaa

=== Code Execution Successful ===
```

Practical no 6: Diffie-Hellman Key Agreement

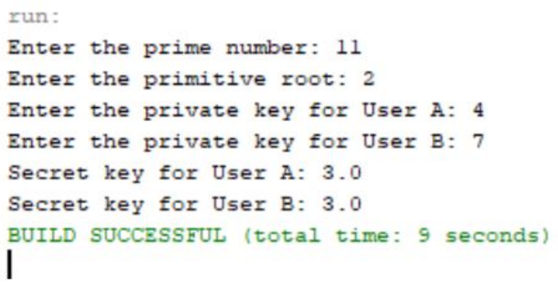
Aim: Write a program to implement the Diffie-Hellman Key Agreement algorithm to generate symmetric keys.

Code:

```
import java.util.*;
public class DiffeHelman {

   static boolean isPrime(double n)
   {
      if (n <= 1)
         return false;

      for (int i = 2; i < n; i++)
         if (n % i == 0)
            return false;

      return true;
   }

   public static void main(String[] args){
      Scanner sc=new
Scanner(System.in);
      System.out.print("Enter the prime
number: ");
      double q=sc.nextDouble();

      Boolean small_check=true;
      Boolean prime_check=true;
      Boolean final_check=true;

      System.out.print("Enter the
primitive root: ");
      double prim_root=sc.nextDouble();
      System.out.print("Enter the private
key for User A: ");
      double
private_key_user_A=sc.nextDouble();

      System.out.print("Enter the private
key for User B: ");
      double
private_key_user_B=sc.nextDouble();

      double public_key_user_A;
      double public_key_user_B;

      double key_user_A;
      double key_user_B;

      if(private_key_user_A<q &&
private_key_user_B<q && prim_root<q
){
         small_check=true;
      }else{
         small_check=false;
      }

      if(isPrime(q)==true &&
isPrime(prim_root)==true){
         prime_check=true;
      }else{
          prime_check=false;
        }

        if(prime_check==false){
          System.out.println("Warning!
Please enter prime values at prime
number and primitive root");
        }
        if(small_check==false){
          System.out.println("Warning!
primitive root and private key should
be smaller than first prime number");
        }

        if(prime_check==true &&
small_check==true){
          final_check=true;
        }else{
          final_check=false;
        }
```

```
        if(final_check==true){

public_key_user_A=(Math.pow(prim_ro
ot,private_key_user_A)%q);

public_key_user_B=(Math.pow(prim_ro
ot,private_key_user_B)%q);

key_user_A=(Math.pow(public_key_user
_B,private_key_user_A)%q);

key_user_B=(Math.pow(public_key_user
_A,private_key_user_B)%q);
           System.out.println("Secret key
for User A: "+key_user_A);
           System.out.println("Secret key
for User B: "+key_user_B);
        }
    }
}
```

Output:



```
run:
Enter the prime number: 11
Enter the primitive root: 2
Enter the private key for User A: 4
Enter the private key for User B: 7
Secret key for User A: 3.0
Secret key for User B: 3.0
BUILD SUCCESSFUL (total time: 9 seconds)
```

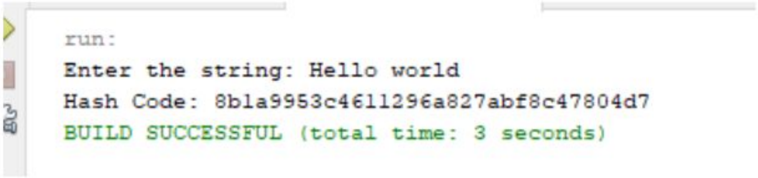
Practical no 7: MD5 Algorithm

Aim: Write a program to implement the MD5 algorithm to compute the message digest.

Code:

```
import java.security.MessageDigest;
import
javax.xml.bind.DatatypeConverter;
import java.util.*;

public class SHAAlgrithm {
  public static String getHash(byte[]
inputBytes,String algorithm){
      String hashvalue="";
      try{
         MessageDigest
message=MessageDigest.getInstance(al
gorithm);
         message.update(inputBytes);
         byte[] digestedByte =
message.digest();

hashvalue=DatatypeConverter.printHex
Binary(digestedByte).toLowerCase();
      }
      catch(Exception e){

      }
      return hashvalue;
   }

   public static void main(String[] args){
      Scanner sc=new
Scanner(System.in);
      System.out.print("Enter the string:
");
      String text=sc.next();
      System.out.println("Hash Code:
"+getHash(text.getBytes(),"MD5"));
   }
}
```

Output:



```
run:
Enter the string: Hello world
Hash Code: 8b1a9953c4611296a827abf8c47804d7
BUILD SUCCESSFUL (total time: 3 seconds)
```