INS Practical

Practical: - Diffie Hellman

```
import java.util.Scanner;
public class DFH {
public static void ComputeDF(double q, double p)
{
//Step 2: Compute CipherKey for Alice and Bob
Scanner sc = new Scanner(System.in);
System.out.println("Enter private key for Alice:");
double pvtAlice = sc.nextInt();
System.out.println("Enter private key for Bob:");
double pvtBob = sc.nextInt();
double cipherKeyAlice, cipherKeyBob;
cipherKeyAlice = Math.pow(p, pvtAlice) % q;
cipherKeyBob = Math.pow(p, pvtBob)% q;
System.out.println("Cipher Key of Alice :"+cipherKeyAlice);
System.out.println("Cipher Key of Bob :"+cipherKeyBob);
//Step 3: Compute Shared Secret Key
double SecretKeyAlice = Math.pow(cipherKeyBob, pvtAlice) %q;
double SecretKeyBob = Math.pow(cipherKeyAlice, pvtBob) %q;
if (SecretKeyAlice == SecretKeyBob)
{
System.out.println("Shared Secret Key = " + (int)SecretKeyAlice);
}
else
System.out.println("Your values don't match. Please try again.");
sc.close();
}
public static boolean checkForPrime(double inputNumber)
{
boolean isItPrime = true;
```

```
if(inputNumber <= 1)
{
isItPrime = false;
return isItPrime;
}
else
{
for (int i = 2; i<= inputNumber/2; i++)
{
if ((inputNumber % i) == 0)
{
isItPrime = false;
break;
}
}
return isItPrime;
}
}
public static void main(String[] args) {
//Step 1 : take q and p as input
Scanner sc = new Scanner(System.in);
System.out.println("Enter value for q(prime no.)- ");
double q;
q = sc.nextInt();
boolean IsPrime = checkForPrime(q);
if (IsPrime)
{
System.out.println("Enter value for p(primitive root of q)- ");
double p;
p = sc.nextInt();
ComputeDF(q, p);
```

}

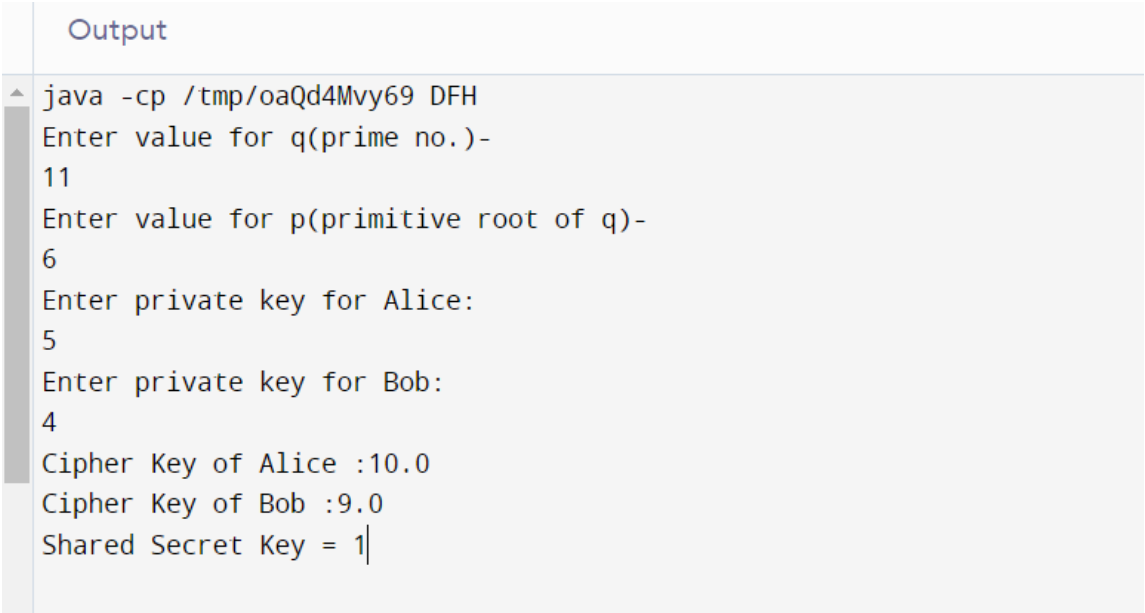
else

System.out.println("The value you entered is not a prime number");

}

}

Output:



Practical: Columnar Technique:

```
import java.util.Scanner;
public class Columnar {
public static void Encrypt(String plainText, String key)
{

String pt[]=plainText.split("");
String ky[]=key.split("");

int columns = key.length();
int rows;
if(plainText.length() % columns == 0)
{
rows = (plainText.length())/columns + 1;
}
else
{
rows = (plainText.length()/columns) + 2;
} //
String[][] mat = new String[rows][columns];
String var="";
int c = 0;
```

```
for(int i = 0; i < columns; i++)
{
mat[0][i] = ky[i];
}

for(int i = 1; i < rows; i++)
{
for(int j = 0; j< columns; j++)
{
if(c != plainText.length())
{
var = pt[c];
mat[i][j]=var;

c++;
}
else
break;
}

}

for(int i = 0; i < rows; i++)
{
for(int j = 0; j< columns; j++)
{
if(mat[i][j] == null)
{
mat[i][j]="X";
}
}
}

for(int i = 0; i < rows; i++)
{
for(int j = 0; j< columns; j++)
{
System.out.print(mat[i][j]+"\t");
}
System.out.println();
}

String output="";
for(int i = 0; i < columns; i++)
{
for(int j = 1; j< rows ; j++)
{
output=output+mat[j][i];
}
}
System.out.println("Encrypted String: "+output);
}
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.println("Enter the String for Encryption: ");
String str = new String();
str = sc.next();
System.out.println("Enter the key for Encryption: ");
```

```
String key = new String();
key = sc.next();
Encrypt(str,key);
}
}
```

Output:

```
Output

java -cp /tmp/c999OutqRa Columnar
Enter the String for Encryption:
abhisheksingh
Enter the key for Encryption:
234
2   3   4   a   b   h
i   s   h
e   k   s
i   n   g
h   X   X
Encrypted String: aieihbsknXhhsgX
```

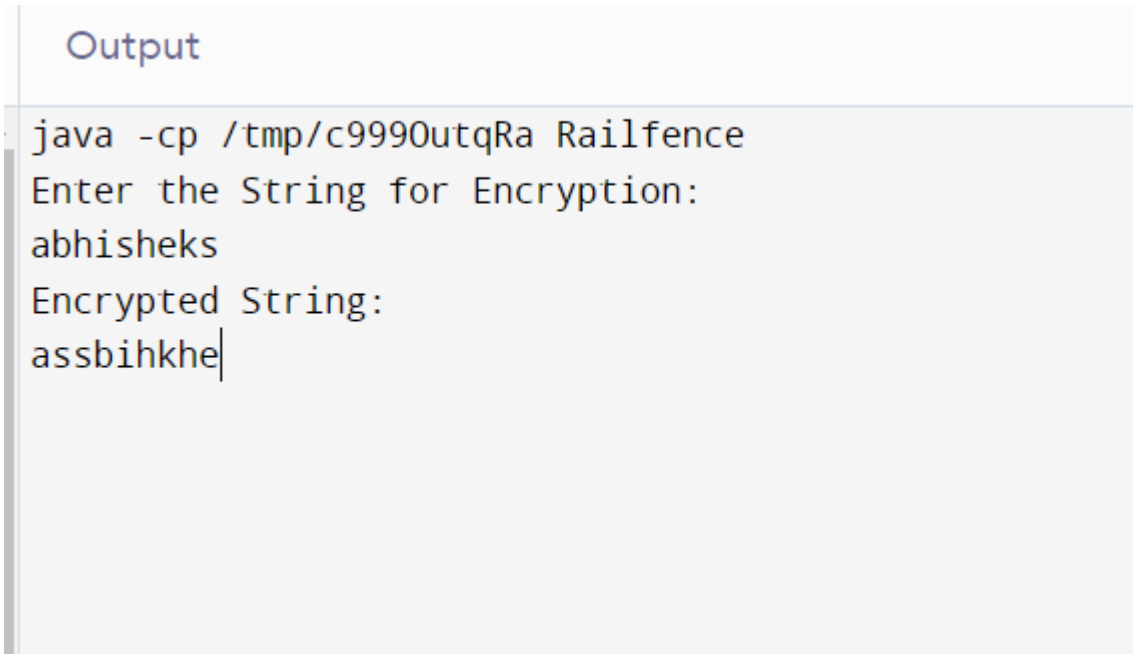
Practical: Railfence

```
import java.util.Arrays;
import java.util.Scanner;
public class Railfence {
public static void Encrypt(String str, int n)
{
//if depth = 1
if (n == 1)
{
System.out.print(str);
return ;
}
char[] str1 = str.toCharArray();
int len = str.length();
String[] arr = new String[n];
Arrays.fill(arr, "");
int row = 0;
boolean down = true;

for (int i = 0; i < len; i++)
{
arr[row] = arr[row] + (str1[i]);
if (row == n - 1)
{
down = false;
}
```

```
else if (row == 0)
{
down = true;
}
if (down)
{
row++;
}
else
{
row--;
}
}
for (int i = 0; i < n; i++)
{
System.out.print(arr[i]);
}
}
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.println("Enter the String for Encryption: ");
String str = new String();
str = sc.next(); //plaintext from user
int n = 3; //key / rows
System.out.println("Encrypted String:");
Encrypt(str, n);
}
}
```

Output



Practical : MD5

Code:

```
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

// Java program to calculate MD5 hash value
public class MD5 {
	public static String getMd5(String input)
	{
		try {

			// Static getInstance method is called with hashing MD5
			MessageDigest md = MessageDigest.getInstance("MD5");

			// digest() method is called to calculate message digest
			// of an input digest() return array of byte
			byte[] messageDigest = md.digest(input.getBytes());

			// Convert byte array into signum representation
			BigInteger no = new BigInteger(1, messageDigest);

			// Convert message digest into hex value
			String hashtext = no.toString(16);
			while (hashtext.length() < 32) {
				hashtext = "0" + hashtext;
			}
			return hashtext;
		}

		// For specifying wrong message digest algorithms
```

```
            catch (NoSuchAlgorithmException e) {
                throw new RuntimeException(e);
            }
        }

        // Driver code
        public static void main(String args[]) throws NoSuchAlgorithmException
        {
            String s = "GeeksForGeeks";
            System.out.println("Your HashCode Generated by MD5 is: " + getMd5(s));
        }
}
```

Output

```
java -cp /tmp/oaQd4Mvy69 MD5
Your HashCode Generated by MD5 is: e39b9c178b2c9be4e99b141d956c6ff6
```

Practical: RSA

```
import java.math.*;
import java.util.*;
class RSA {
        public static void main(String args[])
        {
            int p, q, n, z, d = 0, e, i;
            // The number to be encrypted and decrypted
            int msg = 12;
            double c;
            BigInteger msgback;
```

```
// 1st prime number p
p = 3;
// 2nd prime number q
q = 11;
n = p * q;
z = (p - 1) * (q - 1);
System.out.println("the value of z = " + z);
for (e = 2; e < z; e++) {
        // e is for public key exponent
        if (gcd(e, z) == 1) {
                break;
        }
}
System.out.println("the value of e = " + e);
for (i = 0; i <= 9; i++) {
        int x = 1 + (i * z);
        // d is for private key exponent
        if (x % e == 0) {
                d = x / e;
                break;
        }
}
System.out.println("the value of d = " + d);
c = (Math.pow(msg, e)) % n;
System.out.println("Encrypted message is : " + c);
// converting int value of n to BigInteger
BigInteger N = BigInteger.valueOf(n);
// converting float value of c to BigInteger
BigInteger C = BigDecimal.valueOf(c).toBigInteger();
msgback = (C.pow(d)).mod(N);
System.out.println("Decrypted message is : "
```

```
                                                    + msgback);
        }
        static int gcd(int e, int z)
        {
                if (e == 0)
                        return z;
                else
                        return gcd(z % e, e);
        }
}
```

Output:

Output

```
java -cp /tmp/oaQd4Mvy69 RSA
the value of z = 20
the value of e = 3
the value of d = 7
Encrypted message is : 12.0
Decrypted message is : 12
```