11. Write a program for simple RSA algorithm to encrypt and decrypt the data.

RSA is an example of public key cryptography. It was developed by Rivest, Shamir and Adelman. The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers. The RSA algorithm's efficiency requires a fast method for performing the modular exponentiation operation. A less efficient, conventional method includes raising a number (the input) to a power (the secret or public key of the algorithm, denoted *e* and *d*, respectively) and taking the remainder of the division with *N*. A straight-forward implementation performs these two steps of the operation sequentially: first, raise it to the power and second, apply modulo. The RSA algorithm comprises of three steps, which are depicted below:

Key Generation Algorithm
1. Generate two large random primes, p and q, of approximately equal size such that their product n = p*q
2. Compute n = p*q and Euler‘s totient function (φ) phi(n) = (p-1)(q-1).
3. Choose an integer e, 1 < e < phi, such that gcd(e, phi) = 1.
4. Compute the secret exponent d, 1 < d < phi, such that e*d ≡ 1 (mod phi).
5. The public key is (e, n) and the private key is (d, n). The values of p, q, and phi should also be kept secret.

Encryption
Sender A does the following:-
1. Using the public key (e,n)
2. Represents the plaintext message as a positive integer M
3. Computes the cipher text C = M^e mod n. 4. Sends the cipher text C to B (Receiver).

Decryption
Recipient B does the following:-
1. Uses his private key (d, n) to compute M = C^d mod n.
2. Extracts the plaintext from the integer representative m.

Source Code:

```
import java.util.*;
import java.io.*;
class rsa
{
	static int mult(int x,int y,int n)
	{
		int k=1;
		int j;
		for (j=1; j<=y; j++)
			k = (k * x) % n;
		return ( int) k;
	}
	public static void main (String arg[]) throws Exception
	{
```

```
                Scanner s=new Scanner(System.in);
                InputStreamReader r=new InputStreamReader(System.in);
                BufferedReader br=new BufferedReader(r);
                String msg1;
                int pt[]=new int[100];
                int ct[]=new int[100];
                int n, d, e,Z, p, q, i;
                System.out.println("Enter prime No.s p,q :");
                p=s.nextInt();
                q=s.nextInt();
                n = p*q;
                Z=(p-1)*(q-1);
                System.out.println("\nSelect e value:");
                e=s.nextInt();
                System.out.printf("Enter message : ");
                msg1=br.readLine();
                char msg[]=msg1.toCharArray();
                for(i=0;i<msg.length;i++)
                        pt[i]=msg[i];
                for(d=1;d<Z;++d)
                        if(((e*d)%Z)==1) break;
                                System.out.println("p="+""+p+"\tq="+q+"\tn="+n+
                                                "\tz="+Z+"\te="+e+"\td="+d);
                                System.out.println("\nCipher Text = ");
                for(i=0; i<msg.length; i++)
                        ct[i] = mult(pt[i], e,n);
                for(i=0; i<msg.length; i++)
                        System.out.print("\t"+ct[i]);
                System.out.println("\nPlain Text = ");
                for(i=0; i<msg.length; i++)
                        pt[i] = mult(ct[i], d,n) ;
                for(i=0; i<msg.length; i++)
                        System.out.print((char)pt[i]);
        }
}
```

Output

```
lab3-20@lab320-Veriton-Series: ~/CN
lab3-20@lab320-Veriton-Series:~/CN$ javac rsa.java
Picked up JAVA_TOOL_OPTIONS: -javaagent:/usr/share/java/jayatanaag.jar
lab3-20@lab320-Veriton-Series:~/CN$ java rsa
Picked up JAVA_TOOL_OPTIONS: -javaagent:/usr/share/java/jayatanaag.jar
Enter prime No.s p,q :
13
11

Select e value:
7
Enter message : Computer Networks Laboratory
p=13    q=11    n=143   z=120   e=7     d=103

Cipher Text =
        89      45      21      18      39      129     62      49      98      78      62
129     37      45      49      68      80      98      54      59      32      45      49
59      129     45      49      121
Plain Text =
Computer Networks Laboratory
lab3-20@lab320-Veriton-Series:~/CN$
```