Task 1a

```java
import java.util.*;
class T1{
    public static void main(String args[]){
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Plain Text:");
        String txt=sc.next();
        System.out.println("Enter key:");
        int key=sc.nextInt();
        txt=encrypt(txt,key);
        System.err.println("Encrypted Text: "+txt);
        txt=decrypt(txt,key);
        System.err.println("Decrypted Text: "+txt);
    }
    static String encrypt(String txt,int key){
        String etxt="";
        for(int i=0;i<txt.length();i++){
            char j=txt.charAt(i);
            if(Character.isUpperCase(j)){
                etxt+=(char)(((j-'A')+key)%26 +'A');
            }
            else etxt+=(char)(((j-'a')+key)%26 +'a');
        }
        return etxt;
    }
    static String decrypt(String txt,int key){
        String etxt="";
        for(int i=0;i<txt.length();i++){
            char j=txt.charAt(i);
            if(Character.isUpperCase(j)){
                etxt+=(char)(((j-'A')-key+26)%26 +'A');
            }
            else etxt+=(char)(((j-'a')-key+26)%26 +'a');
        }
        return etxt;
    }
}
```

Task 1b

```java
import java.util.*;
class T1b{
    static String abc="abcdefghijklmnopqrstuvwxyz";
    public static void main(String args[]){
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Plain Text:");
        String txt=sc.next();
        txt=encrypt(txt);
        System.err.println("Encrypted Text: "+txt);
        txt=decrypt(txt);
        System.err.println("Decrypted Text: "+txt);
    }
    static String encrypt(String txt){
        String etxt="";
        for (int i=0;i<txt.length();i++){
            int a=abc.indexOf(txt.charAt(i));
            etxt+=abc.charAt(25-a);
        }
        return etxt;
    }
    static String decrypt(String txt){
        String etxt="";
        for (int i=0;i<txt.length();i++){
            int a=abc.indexOf(txt.charAt(i));
            etxt+=abc.charAt(25-a);
        }
        return etxt;
    }
}
```

Task 2,3,4,5

```
import javax.crypto.*;
public class T2 {
    private static Cipher encryptCipher;
    private static Cipher decryptCipher;
    public static void main(String[] args) throws Exception{
            KeyGenerator keygenerator = KeyGenerator.getInstance("RC4");
            SecretKey secretKey = keygenerator.generateKey();
            encryptCipher = Cipher.getInstance("RC4");
            encryptCipher.init(Cipher.ENCRYPT_MODE, secretKey);
            byte[] encryptedData = encryptData("Classified Information!");
            decryptCipher = Cipher.getInstance("RC4");
            decryptCipher.init(Cipher.DECRYPT_MODE, secretKey);
            decryptData(encryptedData);
    }
    private static byte[] encryptData(String data) throws Exception {
        System.out.println("Data Before Encryption :" + data);
        byte[] dataToEncrypt = data.getBytes();
        byte[] encryptedData = encryptCipher.doFinal(dataToEncrypt);
        System.out.println("Encryted Data: " + encryptedData);
        return encryptedData;
    }
    private static void decryptData(byte[] data) throws Exception {
        byte[] textDecrypted = decryptCipher.doFinal(data);
        System.out.println("Decryted Data: " + new String(textDecrypted));
    }
}
```

Task 7

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

public class T7 {
    private BigInteger p;
    private BigInteger q;
    private BigInteger N;
    private BigInteger phi;
    private BigInteger e;
    private BigInteger d;
    private int bitlength = 1024;
    private Random r;

    public T7() {
        r = new Random();
        p = BigInteger.probablePrime(bitlength, r);
        q = BigInteger.probablePrime(bitlength, r);
        N = p.multiply(q);
        phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
        e = BigInteger.probablePrime(bitlength / 2, r);

        while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0) {
            e = e.add(BigInteger.ONE);
        }

        d = e.modInverse(phi);
    }

    public static void main(String[] args) throws IOException {
        T7 rsa = new T7();
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        String testString;
        System.out.println("Enter the plain text:");
        testString = in.readLine();

        byte[] encrypted = rsa.encrypt(testString.getBytes());
        System.out.println("Encrypted String in Bytes: " + encrypted);

        byte[] decrypted = rsa.decrypt(encrypted);
        System.out.println("Decrypted String: " + new String(decrypted));
    }

    public byte[] encrypt(byte[] message) {
        return (new BigInteger(message)).modPow(e, N).toByteArray();
    }

    public byte[] decrypt(byte[] message) {
        return (new BigInteger(message)).modPow(d, N).toByteArray();
    }

}
```

Task 8

```
import java.io.*;
import java.math.BigInteger;
public class T8
{
public static void main(String[]args)throws IOException
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter prime number:");
BigInteger p=new BigInteger(br.readLine());
System.out.print("Enter primitive root of "+p+":");
BigInteger g=new BigInteger(br.readLine());
System.out.println("Enter value for x less than " +p+":");
BigInteger x=new BigInteger(br.readLine());
BigInteger R1=g.modPow(x,p);
System.out.println("R1="+R1);
System.out.print("Enter value for y less than "+p+":");
BigInteger y=new BigInteger(br.readLine());
BigInteger R2=g.modPow(y,p);
System.out.println("R2="+R2);
BigInteger k1=R2.modPow(x,p);
System.out.println("Key1 calculated :"+k1);
BigInteger k2=R1.modPow(y,p);
System.out.println("Key2 calculated :"+k2);
System.out.println("deffie hellman secret key Encryption has Taken");
}
}
```

Task 9,10

```
import java.security.*;

public class T9 {
    public static void main(String[] args) {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA1");

            System.out.println("Message digest object info:");
            System.out.println("Algorithm = " + md.getAlgorithm());
            System.out.println("ToString = " + md.toString());

            String input;

            input = "";
            md.update(input.getBytes());
            byte[] output = md.digest();
            System.out.println("\nSHA1(\"" + input + "\") = " + bytesToHex(output));
            input = "abc";
            md.update(input.getBytes());
            output = md.digest();
            System.out.println("\nSHA1(\"" + input + "\") = " + bytesToHex(output));

            input = "abcdefghijklmnopqrstuvwxyz";
            md.update(input.getBytes());
            output = md.digest();
            System.out.println("\nSHA1(\"" + input + "\") = " + bytesToHex(output));

        } catch (Exception e) {
            System.out.println("Exception: " + e);
        }
    }

    public static String bytesToHex(byte[] b) {
        char hexDigit[] = {
            '0', '1', '2', '3', '4', '5', '6', '7',
            '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'
        };
        StringBuffer buf = new StringBuffer();
        for (int j = 0; j < b.length; j++) {
            buf.append(hexDigit[(b[j] >> 4) & 0x0f]);
            buf.append(hexDigit[b[j] & 0x0f]);
        }
        return buf.toString();
    }

}
```

Task 12

```
import java.security.*;
import java.util.Base64;

public class T12 {
    public static void main(String[] args) throws Exception {
        KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");
        kpg.initialize(1024);
        KeyPair keyPair = kpg.genKeyPair();

        byte[] data = "Sample Text".getBytes();

        Signature sig = Signature.getInstance("MD5WithRSA");
        sig.initSign(keyPair.getPrivate());
        sig.update(data);

        byte[] signatureBytes = sig.sign();
        System.out.println("Signature: \n" + Base64.getEncoder().encodeToString(signatureBytes));

        sig.initVerify(keyPair.getPublic());
        sig.update(data);
        System.out.println("Signature verification result: " + sig.verify(signatureBytes));
    }
}
```