=======================1. Write a program to perform encryption and decryption using Hill Cipher===============================

```
import java.util.*;

public class HillCipherMini {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // KEY MATRIX INPUT
        System.out.println("Enter 2x2 Key Matrix:");
        int a = sc.nextInt(), b = sc.nextInt();
        int c = sc.nextInt(), d = sc.nextInt();

        sc.nextLine();
        System.out.print("Enter plaintext(2 letters only): ");
        String p = sc.nextLine().toLowerCase();

        int p1 = p.charAt(0) - ’a’;
        int p2 = p.charAt(1) - ’a’;

        // ENCRYPT
        int e1 = (a*p1 + b*p2) % 26;
        int e2 = (c*p1 + d*p2) % 26;

        String enc = "" + (char)(e1 + ’a’) + (char)(e2 + ’a’);
        System.out.println("Encrypted Text: " + enc);

        // FIND INVERSE KEY
        int det = (a*d - b*c) % 26;
        if (det < 0) det += 26;

        int invDet = 0;
        for (int i = 1; i < 26; i++)
            if ((det * i) % 26 == 1) invDet = i;

        // INVERSE MATRIX (adjoint × invDet)
        int ia = ( d * invDet) % 26;
        int ib = (-b * invDet) % 26;
        int ic = (-c * invDet) % 26;
        int id = ( a * invDet) % 26;

        if (ib < 0) ib += 26;
        if (ic < 0) ic += 26;

        // DECRYPT
        int d1 = (ia*e1 + ib*e2) % 26;
        int d2 = (ic*e1 + id*e2) % 26;

        if (d1 < 0) d1 += 26;
        if (d2 < 0) d2 += 26;

        String dec = "" + (char)(d1 + ’a’) + (char)(d2 + ’a’);
        System.out.println("Decrypted Text: " + dec);
    }
}
```

```
Enter 2x2 Key Matrix:
3 3
2 5
Enter plaintext(2 letters only): hi
Encrypted Text: tc
Decrypted Text: hi
```

==========================2. Write a program to perform encryption and decryption using Ceaser cipher ==========================

```
import java.util.Scanner;

public class CaesarCipher {

    public static String encrypt(String text, int shift) {
        String result = "";
        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (c >= ’A’ && c <= ’Z’) {
                c = (char) ((c - ’A’ + shift) % 26 + ’A’);
            } else if (c >= ’a’ && c <= ’z’) {
                c = (char) ((c - ’a’ + shift) % 26 + ’a’);
            }
            result += c;
        }
        return result;
    }

    public static String decrypt(String text, int shift) {
        String result = "";
        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (c >= ’A’ && c <= ’Z’) {
                c = (char) ((c - ’A’ - shift + 26) % 26 + ’A’);
            } else if (c >= ’a’ && c <= ’z’) {
                c = (char) ((c - ’a’ - shift + 26) % 26 + ’a’);
            }
            result += c;
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter text: ");
        String text = sc.nextLine();

        System.out.print("Enter key value: ");
        int shift = sc.nextInt();
        String encrypted = encrypt(text, shift);
        String decrypted = decrypt(encrypted, shift);

        System.out.println("Encrypted: " + encrypted);
        System.out.println("Decrypted: " + decrypted);
    }
}
```

```
Enter text: hello
Enter key: 3
Encrypted: khoor
Decrypted: hello
```

================3. Write a program to perform encryption and decryption using Substitution cipher
==========================

```
import java.util.Scanner;

public class SubstitutionCipher {

    static String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    static String substitute = "QWERTYUIOPASDFGHJKLZXCVBNM";

    public static String encrypt(String text) {
        String result = "";
        text = text.toUpperCase();

        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            int index = alphabet.indexOf(c);

            if (index != -1)
                result += substitute.charAt(index);
            else
                result += c;
        }
        return result;
    }

    public static String decrypt(String text) {
        String result = "";
        text = text.toUpperCase();

        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            int index = substitute.indexOf(c);

            if (index != -1)
                result += alphabet.charAt(index);
            else
                result += c;
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter text: ");
        String text = sc.nextLine();

        String encrypted = encrypt(text);
        String decrypted = decrypt(encrypted);

        System.out.println("Encrypted: " + encrypted);
        System.out.println("Decrypted: " + decrypted);
    }
}
```

```
Enter text: hello
Encrypted: ITSSG
Decrypted: HELLO
```

===============================4. Write a program to implement the DES algorithm.==============================

```
import java.util.*;
import javax.crypto.*;
import javax.crypto.spec.DESKeySpec;

public class SimpleDES {

    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter message to encrypt: ");
        String msg = sc.nextLine();

        System.out.print("Enter 8-byte key: ");
        String key = sc.nextLine();

        if (key.length() != 8) {
            System.out.println("Key must be exactly 8 characters!");
            return;
        }

        DESKeySpec dks = new DESKeySpec(key.getBytes());
        SecretKey keyObj = SecretKeyFactory.getInstance("DES").generateSecret(dks);
        Cipher c = Cipher.getInstance("DES");

        c.init(Cipher.ENCRYPT_MODE, keyObj);
        byte[] enc = c.doFinal(msg.getBytes());

        c.init(Cipher.DECRYPT_MODE, keyObj);
        byte[] dec = c.doFinal(enc);
        System.out.println("Message: " + msg);
        System.out.println("Encrypted: " + Arrays.toString(enc));
        System.out.println("Decrypted: " + new String(dec));
    }
}
```

```
Enter message to encrypt: hello
Enter 8-byte key: mysecret
Message: hello
Encrypted: [12, -90, 101, 97, -68, -53, -107, 40]
Decrypted: hello
```

================================5. Write a program to implement the Blowfish algorithm. ===============

```
import java.util.*;
import javax.crypto.*;
import javax.crypto.spec.SecretKeySpec;

public class SimpleBlowfish {
    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter message: ");
        String msg = sc.nextLine();

        System.out.print("Enter key: ");
        String key = sc.nextLine();

        SecretKeySpec sk = new SecretKeySpec(key.getBytes(), "Blowfish");

        Cipher c = Cipher.getInstance("Blowfish");
        c.init(Cipher.ENCRYPT_MODE, sk);
        byte[] enc = c.doFinal(msg.getBytes());

        c.init(Cipher.DECRYPT_MODE, sk);
        byte[] dec = c.doFinal(enc);

        System.out.println("Message: " + msg);
        System.out.println("Encrypted: " + Arrays.toString(enc));
        System.out.println("Decrypted: " + new String(dec));
    }
}
```

```
Enter message: hello
Enter key: mysecret
Message: hello
Encrypted: [-72, 23, -9, -123, 1, 44, 82, -102]
Decrypted: hello
```

==============================6. Write the RC4 logic in Java Using Java cryptography==============================

```
import javax.crypto.*;
import java.util.Base64;

public class SimpleRC4 {
    public static void main(String[] args) throws Exception {

        KeyGenerator kg = KeyGenerator.getInstance("RC4");
        SecretKey key = kg.generateKey();

        String keyStr = Base64.getEncoder().encodeToString(key.getEncoded());
        System.out.println("Generated Secret Key (Base64): " + keyStr);

        Cipher c = Cipher.getInstance("RC4");
        c.init(Cipher.ENCRYPT_MODE, key);

        String text = "HELLO WORLD";
        byte[] enc = c.doFinal(text.getBytes());
        String encBase = Base64.getEncoder().encodeToString(enc);

        System.out.println("Encrypted Text: " + encBase);

        c.init(Cipher.DECRYPT_MODE, key);
        String dec = new String(c.doFinal(Base64.getDecoder().decode(encBase)));

        System.out.println("Decrypted Text: " + dec);
    }
}
```

```
Generated Secret Key (Base64): knJGLedlMgHLLkVhIfVwKQ==
Encrypted Text: tSkCQdBMNgy+o7I=
Decrypted Text: HELLO WORLD
```

==================================7. Write a program to implement the RSA algorithm.
=====================================

```
import java.security.*;
import javax.crypto.Cipher;
import java.util.Base64;

public class SimpleRSA {
    public static void main(String[] args) throws Exception {

        KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");
        kpg.initialize(2048);

        KeyPair kp = kpg.generateKeyPair();
        PublicKey pub = kp.getPublic();
        PrivateKey prv = kp.getPrivate();

        String msg = "Hello";
        System.out.println("Original Message: " + msg);

        Cipher c = Cipher.getInstance("RSA");
        c.init(Cipher.ENCRYPT_MODE, pub);
        String enc = Base64.getEncoder().encodeToString(c.doFinal(msg.getBytes()));
        System.out.println("Encrypted Message: " + enc);

        c.init(Cipher.DECRYPT_MODE, prv);
        String dec = new String(c.doFinal(Base64.getDecoder().decode(enc)));

        System.out.println("Decrypted Message: " + dec);
    }
}
```

```
Original Message: Hello
Encrypted Message:
mPLVvMdr54n5+BMUYl8hYq5Ol7HsrFK5A80kEqo9KLynYko/iafFMBEnq1NyGDDSjt2hV39U7leBInRg4V
xsr5KU88Lwug0LuWPkZJ0v/eJ1Pk43cNB+ceGXNTQMuI7lR/YWQVTsK5SxAk5Qnv57y4ffFEmW67kbzk
v8z2EcBe/ee42pOcl6M8a2g3++3D7YVad8pQiP9ee+GkTGMryat6NoPo/0QfkmIkWU21+z3kLgLKVb1jDw
/HhGqiwxm168oqq5DR+iFLl4oFjgMboiUHf2ks3PgaLWmG5AqamdL8ZOidJtAk6b3N113o2RLiv4/62oSxN
Ay2R026ViPFTkXA==
Decrypted Message: Hello
```

==============================8. Write a program to Implement the Diffie-Hellman Key Exchange mechanism ========================

```
import java.math.BigInteger;
import java.security.SecureRandom;

public class SimpleDH {
    public static void main(String[] args) {

        BigInteger P = new BigInteger("11223344556677889900998877665544332211", 16);
        BigInteger G = BigInteger.valueOf(2);

        SecureRandom rnd = new SecureRandom();

        BigInteger a = new BigInteger(P.bitLength(), rnd);
        BigInteger A = G.modPow(a, P);

        BigInteger b = new BigInteger(P.bitLength(), rnd);
        BigInteger B = G.modPow(b, P);

        System.out.println("Prime (P): " + P);
        System.out.println("Generator (G): " + G);
        System.out.println("Alice Public Key A = " + A);
        System.out.println("Bob Public Key B = " + B);

        BigInteger KA = B.modPow(a, P);
        BigInteger KB = A.modPow(b, P);

        System.out.println("Shared Key computed by Alice = " + KA);
        System.out.println("Shared Key computed by Bob = " + KB);
        System.out.println("Keys Match? " + KA.equals(KB));
    }
}
```

```
Prime (P): 382091931328340632209310247335727530352255505
Generator (G): 2
```

```
Alice Public Key A = 35124337424571205118003808280717099378823878
Bob Public Key B = 31946730152253868085670668698827356828169511 6
Shared Key computed by Alice = 28949464965000633747075663423116728407447270 1
Shared Key computed by Bob = 28949464965000633747075663423116728407447270 1
Keys Match? true
```

=================9. Calculate the message digest of a text using the SHA-1 algorithm in JAVA.
=================================

```
import java.security.MessageDigest;
import java.util.Scanner;
import java.util.Base64;

public class SimpleSHA1 {

    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter message: ");
        String text = sc.nextLine();

        MessageDigest md = MessageDigest.getInstance("SHA-1");
        byte[] hash = md.digest(text.getBytes());

        StringBuilder hex = new StringBuilder();
        for (byte b : hash)
            hex.append(String.format("%02x", b));

        System.out.println("Digest Hex: " + hex);
        System.out.println("Digest Base64: " + Base64.getEncoder().encodeToString(hash));
    }
}
```

```
Enter message: hello
Digest Hex: aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d
Digest Base64: qvTGHdzF6KLavt4PO0gs2a6pQ00=
```

===========================10. Calculate the message digest of a text using the MD5 algorithm in JAVA=====================

```
import java.security.MessageDigest;
import java.util.Scanner;
import java.util.Base64;

public class SimpleMD5 {

    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter message: ");
        String text = sc.nextLine();

        MessageDigest md = MessageDigest.getInstance("MD5");
        byte[] hash = md.digest(text.getBytes());
        StringBuilder hex = new StringBuilder();
        for (byte b : hash)
            hex.append(String.format("%02x", b));

        System.out.println("Digest Hex: " + hex);
        System.out.println("Digest Base64: " + Base64.getEncoder().encodeToString(hash));
    }
}
```

```
Enter message: hello
Digest Hex: 5d41402abc4b2a76b9719d911017c592
Digest Base64: XUFAKrxLKna5cZ2REBfFkg==

====================================================================================
================================================
```