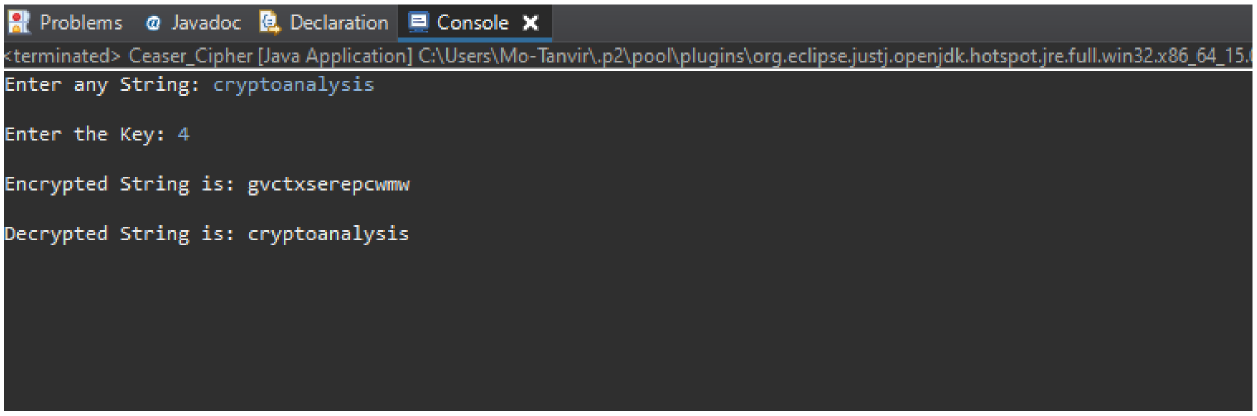
Ceaser Cipher :

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Scanner;
public class Ceaser_Cipher {
	static Scanner sc=new Scanner(System.in);
	static BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
	public static void main(String[] args) throws IOException {
		System.out.print("Enter any String: ");
		String str = br.readLine();
		System.out.print("\nEnter the Key: ");
		int key = sc.nextInt();
		String encrypted = encrypt(str, key);
		System.out.println("\nEncrypted String is: " +encrypted);
		String decrypted = decrypt(encrypted, key);
		System.out.println("\nDecrypted String is: "
		+decrypted); System.out.println("\n");
	}
	public static String encrypt(String str, int key)
	{
		String encrypted = "";
		for(int i = 0; i < str.length(); i++) {
			int c = str.charAt(i);
			if (Character.isUpperCase(c)) {
			c = c + (key % 26);
			if (c > 'Z')
			c = c - 26;
			}
			else if (Character.isLowerCase(c)) {
			c = c + (key % 26);
			if (c > 'z')
			c = c - 26;
			}
			encrypted += (char) c;
		}
		return encrypted;
	}
	public static String decrypt(String str, int key)
	{
		String decrypted = "";
		for(int i = 0; i < str.length(); i++) {
			int c = str.charAt(i);
			if (Character.isUpperCase(c)) {
```

```
                        c = c - (key % 26);
                        if (c < 'A')
                        c = c + 26;
                        }
                        else if (Character.isLowerCase(c)) {
                        c = c - (key % 26);
                        if (c < 'a')
                        c = c + 26;
                        }
                        decrypted += (char) c;
                }
                return decrypted;
        }
}
```

Input/Output:



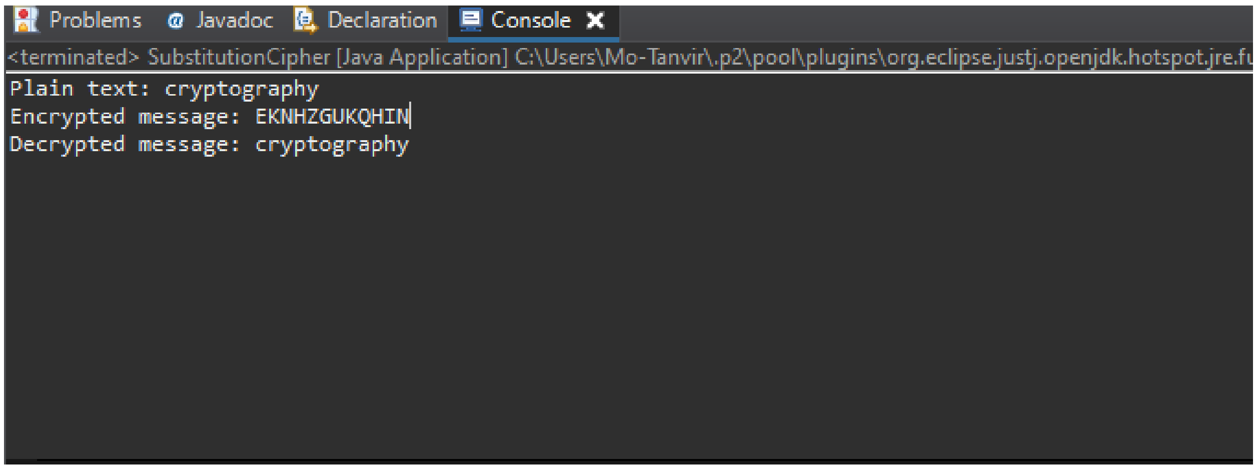
```
<terminated> Ceaser_Cipher [Java Application] C:\Users\Mo-Tanvir\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.
Enter any String: cryptoanalysis

Enter the Key: 4

Encrypted String is: gvctxserepcwmw

Decrypted String is: cryptoanalysis
```

SubstitutionCipher(MonoAlphabetic):

```
import java.io.*;
import java.util.*;
public class SubstitutionCipher {
        static Scanner sc = new Scanner(System.in);
        static BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        public static char normalChar[]
      = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
        'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r',
        's', 't', 'u', 'v', 'w', 'x', 'y', 'z' };

   public static char codedChar[]
      = { 'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O',
        'P', 'A', 'S', 'D', 'F', 'G', 'H', 'J', 'K',
        'L', 'Z', 'X', 'C', 'V', 'B', 'N', 'M' };
        public static void main(String args[])
   {
      String str = "cryptography";
      System.out.println("Plain text: " + str);
      String encryptedString = stringEncryption(str.toLowerCase());
      System.out.println("Encrypted message: "
                      + encryptedString);
      System.out.println("Decrypted message: "
         + stringDecryption(encryptedString));
   }
   public static String stringEncryption(String s)
   {
      String encryptedString = "";
      for (int i = 0; i < s.length(); i++) {
         for (int j = 0; j < 26; j++) {
            if (s.charAt(i) == normalChar[j])
            {
               encryptedString += codedChar[j];
               break;
            }
            if (s.charAt(i) < 'a' || s.charAt(i) > 'z')
            {
               encryptedString += s.charAt(i);
               break;
            }
         }
      }
      return encryptedString;
```

```
    }
    public static String stringDecryption(String s)
    {
        String decryptedString = "";
        for (int i = 0; i < s.length(); i++)
        {
            for (int j = 0; j < 26; j++) {
                if (s.charAt(i) == codedChar[j])
                {
                    decryptedString += normalChar[j];
                    break;
                }
                if (s.charAt(i) < 'A' || s.charAt(i) > 'Z')
                {
                    decryptedString += s.charAt(i);
                    break;
                }
            }
        }
        return decryptedString;
    }
}
```

Input/Output:



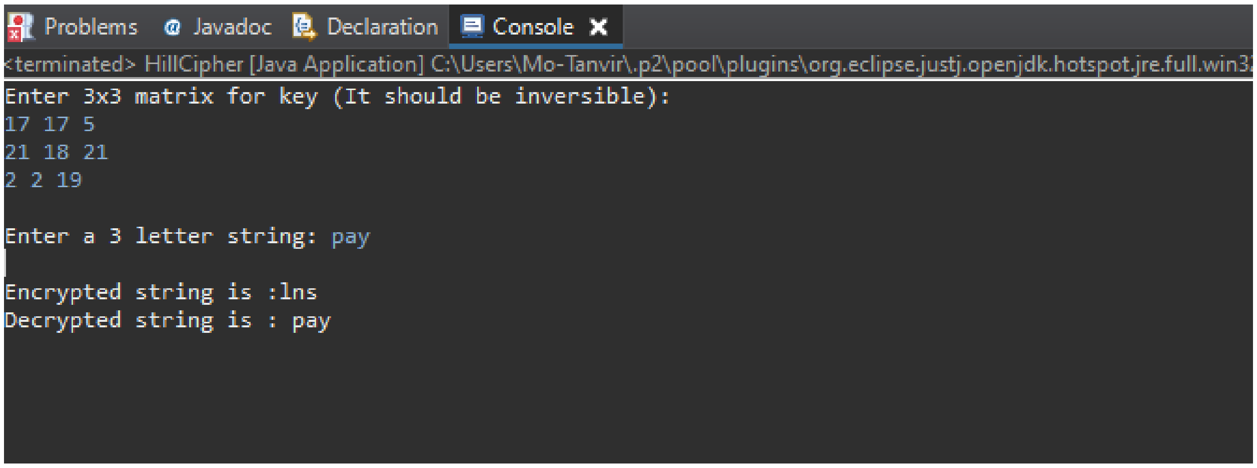
```
<terminated> SubstitutionCipher [Java Application] C:\Users\Mo-Tanvir\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.fu
Plain text: cryptography
Encrypted message: EKNHZGUKQHIN
Decrypted message: cryptography
```

Hillcipher:

```
import java.io.*;
import java.util.*;
import java.io.*;
public class HillCipher {
	static final int N = 3;
	static double[][] decrypt = new double[3][1];
	static double[][] a = new double[3][3];
	static double[][] b = new double[3][3];
	static double[][] mes = new double[3][1];
	static double[][] res = new double[3][1];
	static BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
	static Scanner sc = new Scanner(System.in);
	public static void main(String[] args) throws IOException {
		getkeymes();
		if(inverse()==false){
			System.out.println("Matrix does not have inverse.\nKey is
invalid.");
			return;
		}
		for(int i=0;i<3;i++) {
			for(int j=0;j<1;j++){
				for(int k=0;k<3;k++) {
					res[i][j]=res[i][j]+a[i][k]*mes[k][j];
				}
			}
		}

		System.out.print("\nEncrypted string is :");
		for(int i=0;i<3;i++) {
			System.out.print((char)(res[i][0]%26+97));
			res[i][0]=res[i][0];
		}
		for(int i=0;i<3;i++){
			for(int j=0;j<1;j++){
				for(int k=0;k<3;k++) {
					decrypt[i][j] = decrypt[i][j]+b[i][k]*res[k][j];
				}
			}
		}
		System.out.print("\nDecrypted string is : ");
		for(int i=0;i<3;i++){
			System.out.print((char)(decrypt[i][0]%26+'a'));
		}
		System.out.print("\n");
	}
	public static void getkeymes() throws IOException {
```

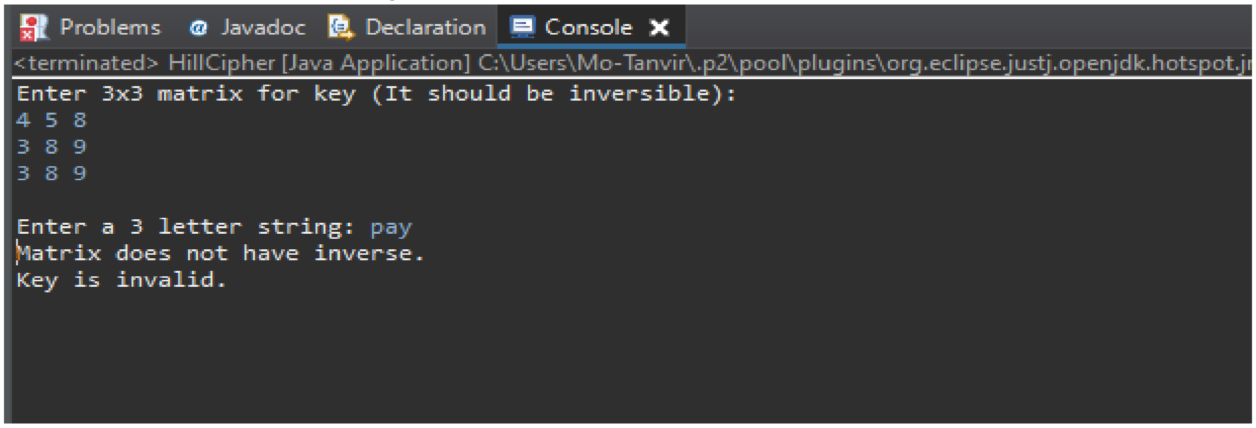
```
            System.out.println("Enter 3x3 matrix for key (It should be inversible): ");
            for(int i=0;i<3;i++){
                  for(int j=0;j<3;j++){
                        a[i][j] = sc.nextDouble();
                  }
            }
            System.out.print("\nEnter a 3 letter string: ");
            String msg = br.readLine();
            for(int i=0;i<3;i++){
                  mes[i][0] = msg.charAt(i)-97;
            }
      }
      public static boolean inverse(){
            for(int i=0;i<N;i++){
                  for(int j=0;j<N;j++){
                        b[j][i]=getCofactor(i,j);
                        if((i+j)%2==1) b[j][i]*=-1;
                  }
            }
            double D=determinant();
            if(Double.compare(D,0.0d)==0){
                  return false;
            }
            else{
                  for(int i=0;i<N;i++){
                        for(int j=0;j<N;j++){
                              b[i][j]/=D;
                        }
                  }
                  return true;
            }
      }
      static double getCofactor(int p, int q)
      {
         int i = 0, j = 0;
         double[][] temp= new double[2][2];
         for (int row = 0; row < N; row++)
         {
            for (int col = 0; col < N; col++)
            {
               if (row != p && col != q)
               {
                  temp[i][j++] = a[row][col];
                  if (j == N - 1)
                  {
                     j = 0;
                     i++;
                  }
               }
            }
            }
            return ((temp[0][0]*temp[1][1])-(temp[0][1]*temp[1][0]));
        }
        static double determinant()
        {
            double D = 0;
            for(int col=0;col<N;col++){
                if(col%2==1){
                        D+=(a[0][col]*getCofactor(0,col)*-1);
                }
                else D+=(a[0][col]*getCofactor(0,col));
            }
            return D;
        }
}
```

Input/Output:



```
<terminated> HillCipher [Java Application] C:\Users\Mo-Tanvir\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win3
Enter 3x3 matrix for key (It should be inversible):
17 17 5
21 18 21
2 2 19

Enter a 3 letter string: pay

Encrypted string is :lns
Decrypted string is : pay
```

In case of non-inversible key:

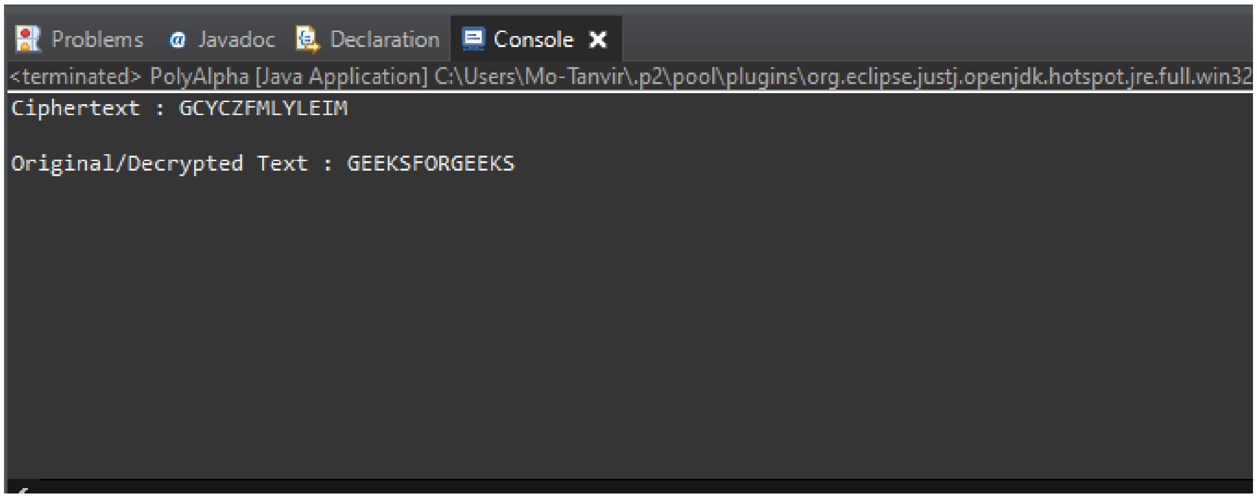


```
<terminated> HillCipher [Java Application] C:\Users\Mo-Tanvir\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jr
Enter 3x3 matrix for key (It should be inversible):
4 5 8
3 8 9
3 8 9

Enter a 3 letter string: pay
Matrix does not have inverse.
Key is invalid.
```

Poly Alphabetic Substitution:

```
class PolyAlpha
{

        static String generateKey(String str, String key)
        {
                int x = str.length();

                for (int i = 0; ; i++)
                {
                        if (x == i)
                                i = 0;
                        if (key.length() == str.length())
                                break;
                        key+=(key.charAt(i));
                }
                return key;
        }
        static String cipherText(String str, String key)
        {
                String cipher_text="";
                for (int i = 0; i < str.length(); i++)
                {
                        int x = (str.charAt(i) + key.charAt(i)) %26;
                        x += 'A';

                        cipher_text+=(char)(x);
                }
                return cipher_text;
        }
        static String originalText(String cipher_text, String key)
        {
                String orig_text="";
                for (int i = 0 ; i < cipher_text.length() &&
                                                        i < key.length(); i++)
                {
                        int x = (cipher_text.charAt(i) -
                                                key.charAt(i) + 26) %26;
                        x += 'A';
                        orig_text+=(char)(x);
                }
                return orig_text;
        }
        static String LowerToUpper(String s)
        {
                StringBuffer str =new StringBuffer(s);
                for(int i = 0; i < s.length(); i++)
                {
                        if(Character.isLowerCase(s.charAt(i)))
                        {
                                str.setCharAt(i, Character.toUpperCase(s.charAt(i)));
                        }
                }
                s = str.toString();
                return s;
        }
        public static void main(String[] args)
        {
                String Str = "GEEKSFORGEEKS";
                String Keyword = "AYUSH";

                String str = LowerToUpper(Str);
                String keyword = LowerToUpper(Keyword);

                String key = generateKey(str, keyword);
                String cipher_text = cipherText(str, key);

                System.out.println("Ciphertext : "
                        + cipher_text + "\n");

                System.out.println("Original/Decrypted Text : "
                        + originalText(cipher_text, key));
        }
}
```

Input/Output:

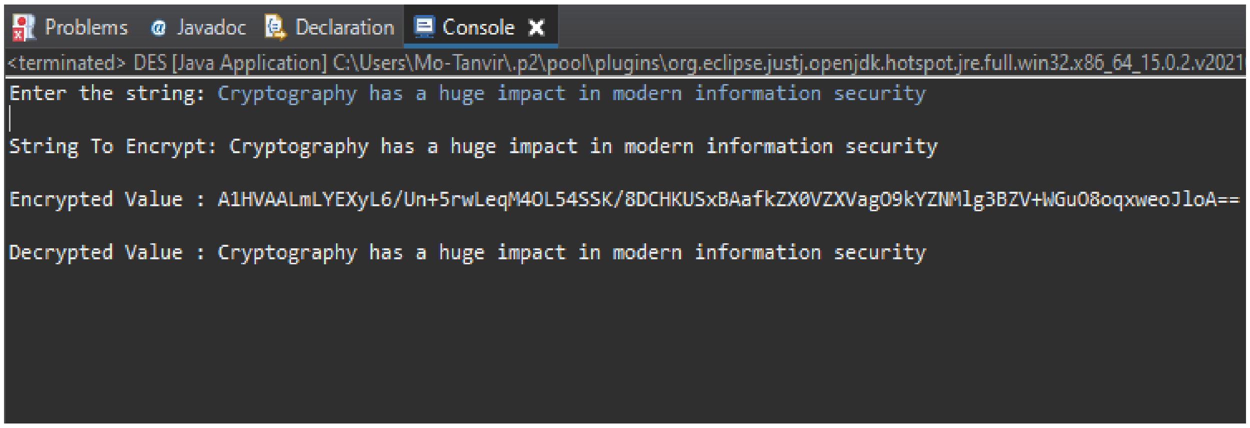


```
<terminated> PolyAlpha [Java Application] C:\Users\Mo-Tanvir\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32
Ciphertext : GCYCZFMLYLEIM

Original/Decrypted Text : GEEKSFORGEEKS
```

DES:

```
import java.util.*;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.security.spec.KeySpec;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESedeKeySpec;
import java.util.Base64;
public class DES {
        private static final String UNICODE_FORMAT = "UTF8";
        public static final String DESEDE_ENCRYPTION_SCHEME = "DESede";
        private KeySpec myKeySpec;
        private SecretKeyFactory mySecretKeyFactory;
        private Cipher cipher;
        byte[] keyAsBytes;
        private String myEncryptionKey;
        private String myEncryptionScheme;
        SecretKey key;
        static BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        public DES() throws Exception {
                myEncryptionKey = "ThisIsSecretEncryptionKey";
                myEncryptionScheme =DESEDE_ENCRYPTION_SCHEME;
                keyAsBytes =myEncryptionKey.getBytes(UNICODE_FORMAT);
                myKeySpec= new DESedeKeySpec(keyAsBytes);
                mySecretKeyFactory =
SecretKeyFactory.getInstance(myEncryptionScheme);
                cipher = Cipher.getInstance(myEncryptionScheme);
                key = mySecretKeyFactory.generateSecret(myKeySpec);
        }
        public String encrypt(String unencryptedString)
        {
                byte[] encryptedString = null;
                try {
                        cipher.init(Cipher.ENCRYPT_MODE, key);
                        byte[] plainText =
unencryptedString.getBytes(UNICODE_FORMAT);
                        byte[] encryptedText = cipher.doFinal(plainText);
                        Base64.Encoder base64encoder = Base64.getEncoder();
                        encryptedString = base64encoder.encode(encryptedText);
                }
                catch (Exception e) {
                        e.printStackTrace();
                }
                return bytes2String(encryptedString);
        }
        public String decrypt(String encryptedString)
        {
                String decryptedText=null;
                try {
                        cipher.init(Cipher.DECRYPT_MODE, key);
                        Base64.Decoder base64decoder = Base64.getDecoder();
                        byte[] encryptedText = base64decoder.decode(encryptedString);
                        byte[] plainText = cipher.doFinal(encryptedText);
                        decryptedText=bytes2String(plainText);
                }
                catch (Exception e) {
                        e.printStackTrace();
                }
                return decryptedText;
        }
        private static String bytes2String(byte[] bytes)
        {
                StringBuffer stringBuffer = new StringBuffer();
                for (int i = 0; i <bytes.length;i++){
                        stringBuffer.append((char) bytes[i]);
                }
                return stringBuffer.toString();
        }
        public static void main(String args []) throws Exception
        {
                System.out.print("Enter the string: ");
                DES myEncryptor= new DES();
                String stringToEncrypt = br.readLine();
                String encrypted = myEncryptor.encrypt(stringToEncrypt); String
decrypted =
                myEncryptor.decrypt(encrypted); System.out.println("\nString To
Encrypt: "
                +stringToEncrypt); System.out.println("\nEncrypted Value : "
+encrypted);
                System.out.println("\nDecrypted Value : " +decrypted);
                System.out.println("");
        }
}
```

Input/Output:

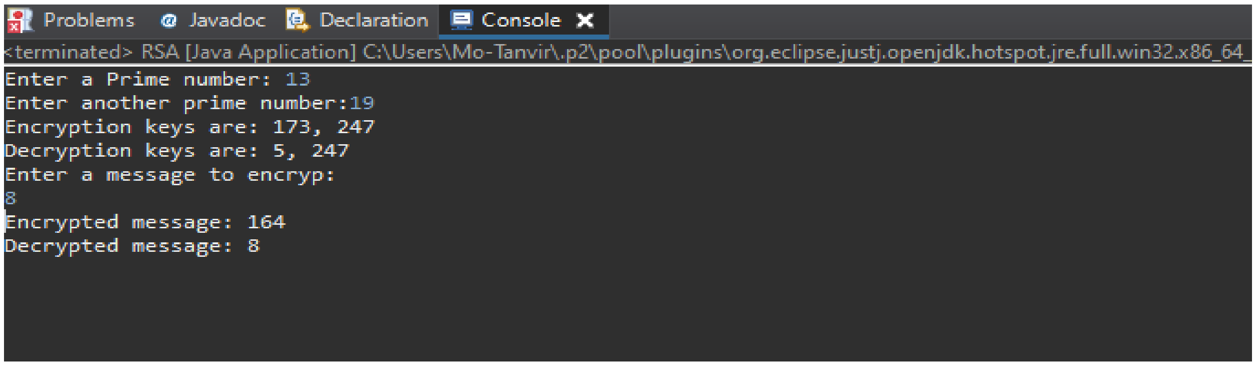


```
<terminated> DES [Java Application] C:\Users\Mo-Tanvir\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v2021
Enter the string: Cryptography has a huge impact in modern information security

String To Encrypt: Cryptography has a huge impact in modern information security

Encrypted Value : A1HVAALmLYEXyL6/Un+5rwLeqM4OL54SSK/8DCHKUSxBAafkZX0VZXVagO9kYZNMlg3BZV+WGuO8oqxweoJloA==

Decrypted Value : Cryptography has a huge impact in modern information security
```

RSA:

```java
import java.io.BufferedReader;
import java.math.*;
import java.util.Random;
import java.util.Scanner;
public class RSA {
        static Scanner sc = new Scanner(System.in);
        public static void main(String[] args) {
                System.out.print("Enter a Prime number: ");
                BigInteger p = sc.nextBigInteger();
                System.out.print("Enter another prime number:");
                BigInteger q = sc.nextBigInteger();
                BigInteger n = p.multiply(q);
                BigInteger phi =
p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
                BigInteger e = generateE(phi);
                BigInteger d = e.modInverse(phi);
                System.out.println("Encryption keys are: " + e + ", " + n);
                System.out.println("Decryption keys are: " + d + ", " + n);
                System.out.println("Enter a message to encryp: ");
                BigInteger m=sc.nextBigInteger();
                BigInteger c=m.modPow(e,n);
                System.out.println("Encrypted message: "+c);
                m=c.modPow(d,n);
                System.out.println("Decrypted message: "+m);
        }
        public static BigInteger generateE(BigInteger fiofn) {
                int y, intGCD;
                BigInteger e;
                BigInteger gcd;
                Random x = new Random();
                do {
                        y = x.nextInt(fiofn.intValue()-1);
                        String z = Integer.toString(y);
                        e = new BigInteger(z);
                        gcd = fiofn.gcd(e);
                        intGCD = gcd.intValue();
                }
                while(y <= 2 || intGCD != 1);
                return e;
        }
}
```

Input/Output:

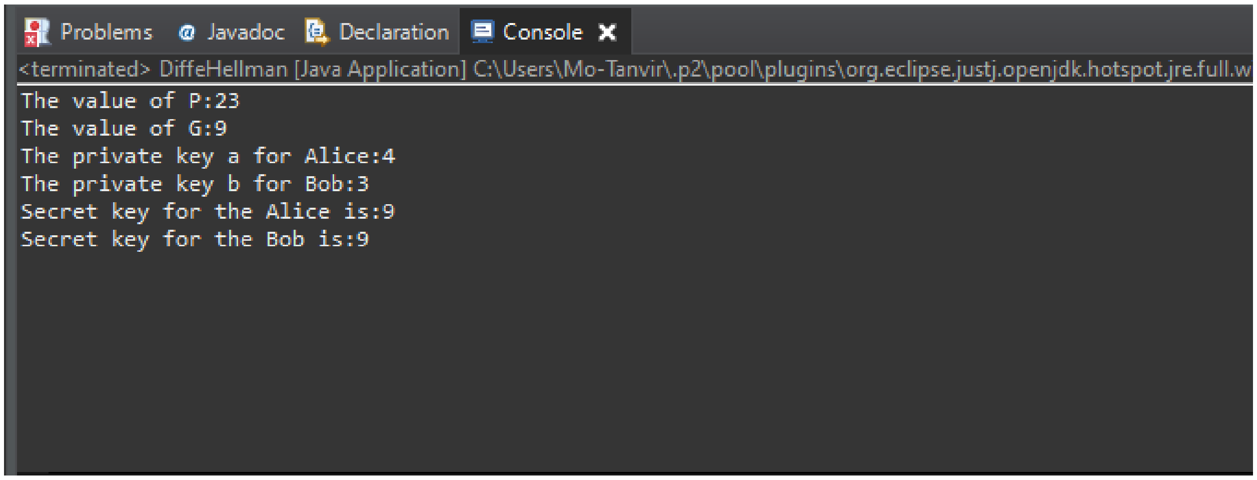


```
<terminated> RSA [Java Application] C:\Users\Mo-Tanvir\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_
Enter a Prime number: 13
Enter another prime number:19
Encryption keys are: 173, 247
Decryption keys are: 5, 247
Enter a message to encryp:
8
Encrypted message: 164
Decrypted message: 8
```

DiffeHellman:

```
import java.math.*;
class DiffeHellman{
        private static long power(long a, long b, long p)
        {
                if (b == 1)
                        return a;
                else
                        return (((long)Math.pow(a, b)) % p);
        }
        public static void main(String[] args)
        {
                long P, G, x, a, y, b, ka, kb;
                P = 23;
                System.out.println("The value of P:" + P);
                G = 9;
                System.out.println("The value of G:" + G);
                a = 4;
                System.out.println("The private key a for Alice:" + a);
                x = power(G, a, P);
                b = 3;
                System.out.println("The private key b for Bob:" + b);
                y = power(G, b, P);
                ka = power(y, a, P);
                kb = power(x, b, P);

                System.out.println("Secret key for the Alice is:" + ka);
                System.out.println("Secret key for the Bob is:" + kb);
        }
}
```

Input/Output:



```
<terminated> DiffeHellman [Java Application] C:\Users\Mo-Tanvir\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.w
The value of P:23
The value of G:9
The private key a for Alice:4
The private key b for Bob:3
Secret key for the Alice is:9
Secret key for the Bob is:9
```