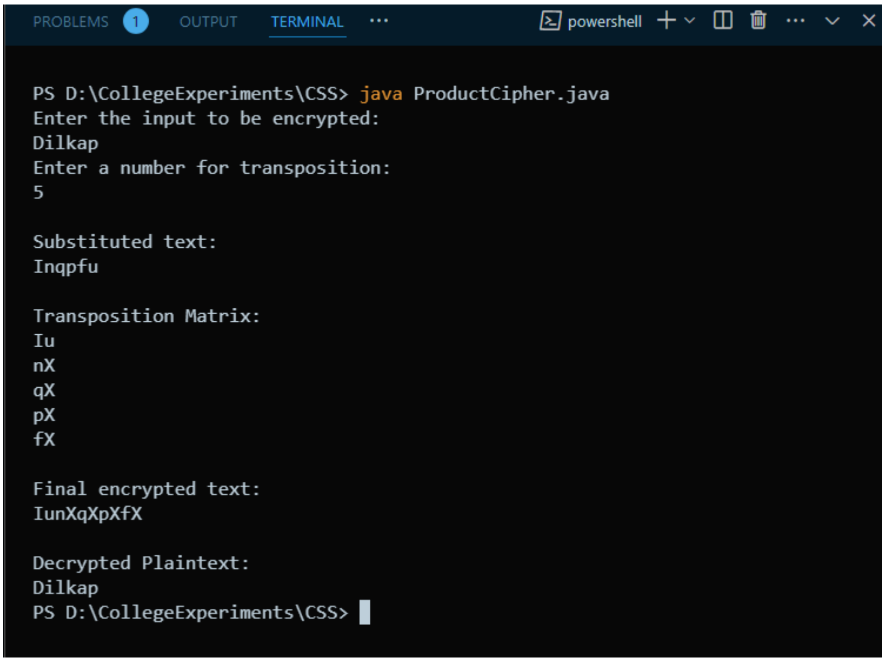
Experiment No. 1

```
import java.util.*;

class ProductCipher {

  public static void main(String args[]) {

    Scanner scanner = new Scanner(System.in);


    // Input for substitution encryption

    System.out.println("Enter the input to be encrypted:");

    String substitutionInput = scanner.nextLine();


    // Input for transposition encryption

    System.out.println("Enter a number for transposition:");

    int n = scanner.nextInt();


    // Substitution encryption

    StringBuffer substitutionOutput = new StringBuffer();

    for (int i = 0; i < substitutionInput.length(); i++) {

      char c = substitutionInput.charAt(i);

      substitutionOutput.append((char) (c + 5)); // Shift each character by 5

    }

    System.out.println("\nSubstituted text:");

    System.out.println(substitutionOutput);


    // Transposition encryption

    String transpositionInput = substitutionOutput.toString();

    int modulus = transpositionInput.length() % n;

    if (modulus != 0) {

      modulus = n - modulus; // Calculate padding needed

      for (; modulus != 0; modulus--) {
```

```
    transpositionInput += "X"; // Add padding character 'X'
  }
}
StringBuffer transpositionOutput = new StringBuffer();
System.out.println("\nTransposition Matrix:");
for (int i = 0; i < n; i++) {
  for (int j = 0; j < transpositionInput.length() / n; j++) {
    char c = transpositionInput.charAt(i + (j * n));
    System.out.print(c); // Print matrix row-wise
    transpositionOutput.append(c);
  }
  System.out.println();
}
System.out.println("\nFinal encrypted text:");
System.out.println(transpositionOutput);

// Transposition decryption
String transpositionEncrypted = transpositionOutput.toString();
int rows = transpositionEncrypted.length() / n;
StringBuffer transpositionPlaintext = new StringBuffer();
for (int i = 0; i < rows; i++) {
  for (int j = 0; j < n; j++) {
    char c = transpositionEncrypted.charAt(i + (j * rows));
    transpositionPlaintext.append(c);
  }
}

// Remove padding
while (transpositionPlaintext.charAt(transpositionPlaintext.length() - 1) == 'X') {
```

```
            transpositionPlaintext.deleteCharAt(transpositionPlaintext.length() - 1);
        }

        // Substitution decryption
        StringBuffer plaintext = new StringBuffer();
        for (int i = 0; i < transpositionPlaintext.length(); i++) {
            char c = transpositionPlaintext.charAt(i);
            plaintext.append((char) (c - 5)); // Reverse shift by 5
        }

        System.out.println("\nDecrypted Plaintext:");
        System.out.println(plaintext);

        scanner.close();
    }
}
```

Output:



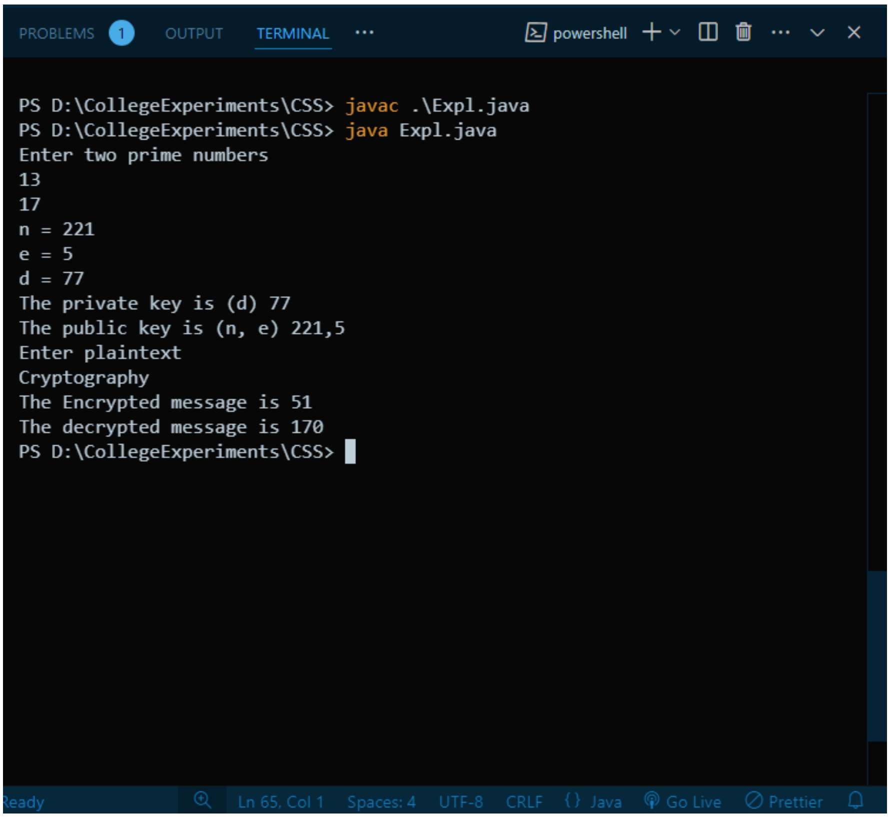
```
PS D:\CollegeExperiments\CSS> java ProductCipher.java
Enter the input to be encrypted:
Dilkap
Enter a number for transposition:
5

Substituted text:
Inqpfu

Transposition Matrix:
Iu
nX
qX
pX
fX

Final encrypted text:
IunXqXpXfX

Decrypted Plaintext:
Dilkap
PS D:\CollegeExperiments\CSS>
```

Experiment No. 2

```
import java.util.*;
class Expl {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int d = 0;

        System.out.println("Enter two prime numbers");
        int p = sc.nextInt();
        int q = sc.nextInt();

        int n = p * q;
        System.out.println("n = " + n);

        int pn = (p - 1) * (q - 1);
        int e = 0;

        search:
        for (int i = 2; i <= pn; i++) {
            int j = i;
            int k = pn;

            while (k != j) {
                if (k > j)
                    k = k - j;
                else
                    j = j - k;
            }
    if (k == 1) {
        e = i;
        break search;
    }
}
System.out.println("e = " + e);

go:
for (int i = 1; i < pn; i++) {
    int x = (e * i) % pn;
    if (x == 1) {
        System.out.println("d = " + i);
        System.out.println("The private key is (d) " + i);
        d = i;
        break go;
    }
}

System.out.println("The public key is (n, e) " + n + "," + e);

System.out.println("Enter plaintext");
String t = sc.next();
int c, m = 0;

for (int i = 0; i < t.length(); i++) {
    m += (int) t.charAt(i);
}
```

```
        c = (m * e) % n;

        System.out.println("The Encrypted message is " + c);


        m = (c * d) % n;

        System.out.println("The decrypted message is " + m);

    }

}
```

Output:



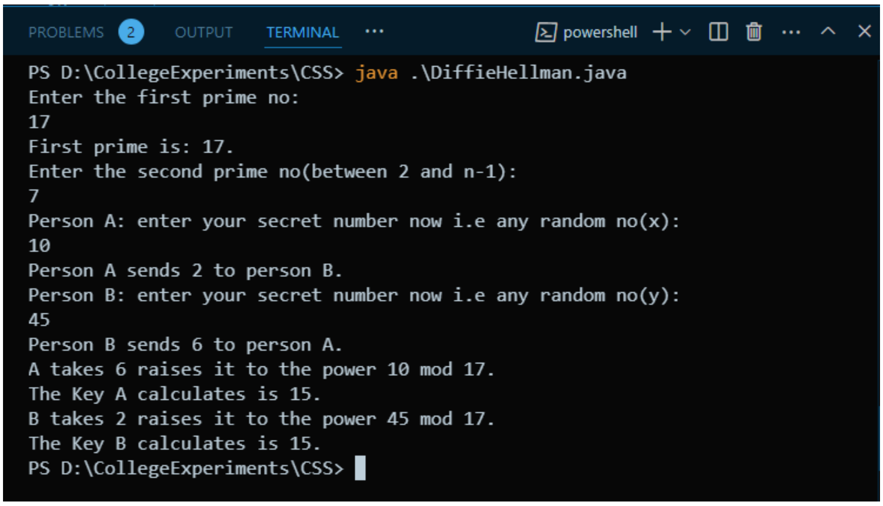
```
PS D:\CollegeExperiments\CSS> javac .\Expl.java
PS D:\CollegeExperiments\CSS> java Expl.java
Enter two prime numbers
13
17
n = 221
e = 5
d = 77
The private key is (d) 77
The public key is (n, e) 221,5
Enter plaintext
Cryptography
The Encrypted message is 51
The decrypted message is 170
PS D:\CollegeExperiments\CSS>
```

Experiment No. 3

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

public class DiffieHellman {
   final static BigInteger one  =  new BigInteger("1");

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

      // Get a start spot to pick a prime from the user.
      System.out.println("Enter the first prime no:");
      String ans = stdin.next();
      n = getNextPrime(ans);

      System.out.println("First prime is: " + n + ".");

      // Get the base for exponentiation from the user.
      System.out.println("Enter the second prime no(between 2 and n-1):");
      BigInteger g = new BigInteger(stdin.next());

      // Get A’s secret number.
      System.out.println("Person A: enter your secret number now i.e any random no(x):");
      BigInteger a = new BigInteger(stdin.next());

      // Make A’s calculation.
      BigInteger resulta = g.modPow(a, n);
```

```
        // This is the value that will get sent from A to B.
        // This value does NOT compromise the value of a easily.
        System.out.println("Person A sends " + resulta + " to person B.");

        // Get B's secret number.
        System.out.println("Person B: enter your secret number now i.e any random no(y):");
        BigInteger b = new BigInteger(stdin.next());

        // Make B's calculation.
        BigInteger resultb = g.modPow(b, n);
        System.out.println("Person B sends " + resultb + " to person A.");

        // Key A calculates
        BigInteger KeyACalculates = resultb.modPow(a, n);
        // Key B calculates
        BigInteger KeyBCalculates = resulta.modPow(b, n);

        // Print out the Key A calculates.
        System.out.println("A takes " + resultb + " raises it to the power " + a + " mod " + n +
    ".");
        System.out.println("The Key A calculates is " + KeyACalculates + ".");

        // Print out the Key B calculates.
        System.out.println("B takes " + resulta + " raises it to the power " + b + " mod " + n +
    ".");
        System.out.println("The Key B calculates is " + KeyBCalculates + ".");
      }

      public static BigInteger getNextPrime(String ans) {
        BigInteger test = new BigInteger(ans);
        while (!test.isProbablePrime(99))
            test = test.add(one);
        return test;
    }
}
```

Output:



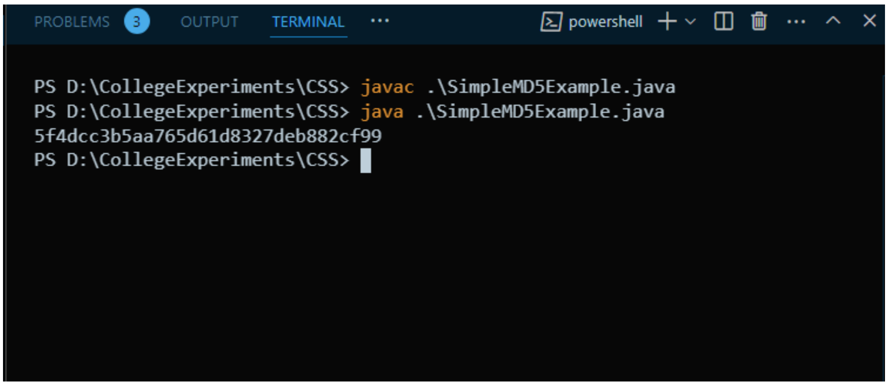
```
PS D:\CollegeExperiments\CSS> java .\DiffieHellman.java
Enter the first prime no:
17
First prime is: 17.
Enter the second prime no(between 2 and n-1):
7
Person A: enter your secret number now i.e any random no(x):
10
Person A sends 2 to person B.
Person B: enter your secret number now i.e any random no(y):
45
Person B sends 6 to person A.
A takes 6 raises it to the power 10 mod 17.
The Key A calculates is 15.
B takes 2 raises it to the power 45 mod 17.
The Key B calculates is 15.
PS D:\CollegeExperiments\CSS>
```

Experiment No. 4

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;

public class SimpleMD5Example {
  public static void main(String[] args) {
    String passwordToHash = "password";
    String generatedPassword = null;

    try {
      // Create MessageDigest instance for MD5
      // For hashing using MD5 can be replaced by SHA1 in the following line
      MessageDigest md = MessageDigest.getInstance("MD5");

      // Add password bytes to digest
      md.update(passwordToHash.getBytes());

      // Get the hash's bytes
      byte[] bytes = md.digest();

      // This bytes[] has bytes in decimal format;
      // Convert it to hexadecimal format
      StringBuilder sb = new StringBuilder();
      for (int i = 0; i < bytes.length; i++) {
        sb.append(Integer.toString((bytes[i] & 0xff) + 0x100, 16).substring(1));
      }
```

```
            // Get complete hashed password in hex format
            generatedPassword = sb.toString();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }

        System.out.println(generatedPassword);
    }
}
```

Output:



```
PS D:\CollegeExperiments\CSS> javac .\SimpleMD5Example.java
PS D:\CollegeExperiments\CSS> java .\SimpleMD5Example.java
5f4dcc3b5aa765d61d8327deb882cf99
PS D:\CollegeExperiments\CSS> 
```