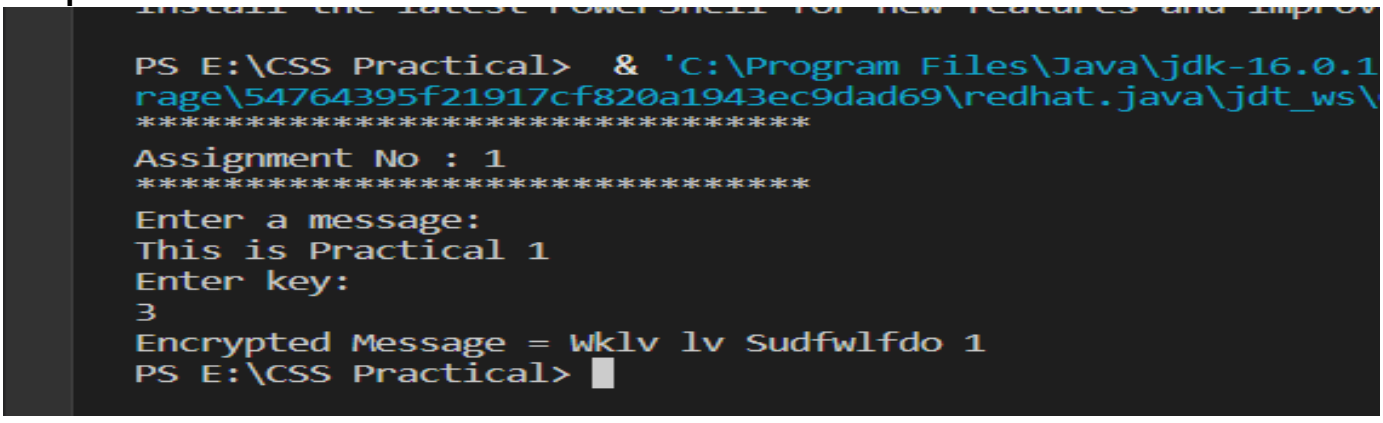
Assignment-1

Code:

```
import java.util.Scanner;
public class P1 {
    public static void main(String... s) {
        String message, encryptedMessage = "";
        int key;
        char ch;
        Scanner sc = new Scanner(System.in);
        System.out.println("*******************************");
        System.out.println("Assignment No : 1");
        System.out.println("*******************************");
        System.out.println("Enter a message: ");
        message = sc.nextLine();
        System.out.println("Enter key: ");
        key = sc.nextInt();
        for (int i = 0; i < message.length();

                ++i) {
            ch = message.charAt(i);
            if (ch >= 'a' && ch <= 'z') {
                ch = (char) (ch + key);
                if (ch > 'z') {
                    ch = (char) (ch - 'z' + 'a' - 1);
                }

                encryptedMessage += ch;

            } else if (ch >= 'A' && ch <= 'Z') {
                ch = (char) (ch + key);
                if (ch > 'Z') {
                    ch = (char) (ch - 'Z' + 'A' - 1);
                }
                encryptedMessage += ch;
            } else {
                encryptedMessage += ch;
            }
        }
        System.out.println("Encrypted Message = " + encryptedMessage);

    }
}
```

Output:



Assignment-2

Code:

```
import java.util.Arrays;
import java.util.Scanner;

public class P2 {
   private static char[][] keySquare;

   private static void generateKeySquare(String key) {
      key = key.replace("J", "I").toUpperCase();
      key = key.replaceAll("[^A-Z]", "");
      String alphabet =
"ABCDEFGHIKLMNOPQRSTUVWXYZ";
      String combinedKey = key + alphabet;
      combinedKey =
combinedKey.replaceAll("(.)(?=.*\\1)", ""); // Remove
duplicate characters
      keySquare = new char[5][5];
      int rowIndex = 0;
      int colIndex = 0;
      for (char ch : combinedKey.toCharArray()) {
         keySquare[rowIndex][colIndex] = ch;
         colIndex++;
         if (colIndex == 5) {
            colIndex = 0;
            rowIndex++;
         }
      }
   }

   private static String preparePlainText(String
plainText) {
      plainText = plainText.replace("J",
"I").toUpperCase();
      plainText = plainText.replaceAll("[^A-Z]", "");
      StringBuilder preparedText = new
StringBuilder(plainText);
      for (int i = 0; i < preparedText.length(); i += 2) {
         if (i + 1 == preparedText.length()) {
            preparedText.append('X');
         } else if (preparedText.charAt(i) ==
preparedText.charAt(i + 1)) {
            preparedText.insert(i + 1, 'X');
         }
      }
      return preparedText.toString();
   }



 private static String encrypt(String plainText) {
      StringBuilder encryptedText = new StringBuilder();
      for (int i = 0; i < plainText.length(); i += 2) {
         char ch1 = plainText.charAt(i);
         char ch2 = plainText.charAt(i + 1);
         int row1 = -1, col1 = -1, row2 = -1, col2 = -1;


         for (int row = 0; row < 5; row++) {
            for (int col = 0; col < 5; col++) {
               if (keySquare[row][col] == ch1) {
                  row1 = row;
                  col1 = col;
               }
               if (keySquare[row][col] == ch2) {
                  row2 = row;
                  col2 = col;
               }
            }
         }
         char encryptedCh1, encryptedCh2;
         if (row1 == row2) {
            encryptedCh1 = keySquare[row1][(col1 + 1) %
5];
            encryptedCh2 = keySquare[row2][(col2 + 1) %
5];
         } else if (col1 == col2) {
            encryptedCh1 = keySquare[(row1 + 1) %
5][col1];
            encryptedCh2 = keySquare[(row2 + 1) %
5][col2];
         } else {
            encryptedCh1 = keySquare[row1][col2];
            encryptedCh2 = keySquare[row2][col1];
         }
         encryptedText.append(encryptedCh1).append(
encryptedCh2);
      }
      return encryptedText.toString();
   }

   private static String decrypt(String encryptedText) {
      StringBuilder decryptedText = new StringBuilder();
      for (int i = 0; i < encryptedText.length(); i += 2) {
         char ch1 = encryptedText.charAt(i);
         char ch2 = encryptedText.charAt(i + 1);
         int row1 = -1, col1 = -1, row2 = -1, col2 = -1;
         for (int row = 0; row < 5; row++) {
            for (int col = 0; col < 5; col++) {
               if (keySquare[row][col] == ch1) {
                  row1 = row;
                  col1 = col;
               }
               if (keySquare[row][col] == ch2) {
                  row2 = row;
                  col2 = col;
```

```
            }
          }
        }
        char decryptedCh1, decryptedCh2;
        if (row1 == row2) {
          decryptedCh1 = keySquare[row1][(col1 + 4) %
5];
          decryptedCh2 = keySquare[row2][(col2 + 4) %
5];
        } else if (col1 == col2) {
          decryptedCh1 = keySquare[(row1 + 4) %
5][col1];
          decryptedCh2 = keySquare[(row2 + 4) %
5][col2];
        } else {
          decryptedCh1 = keySquare[row1][col2];
          decryptedCh2 = keySquare[row2][col1];
        }
        decryptedText.append(decryptedCh1).append(
decryptedCh2);
      }
      return decryptedText.toString();
    }

    public static void main(String[] args) {
        String key = "KEYWORD";
        generateKeySquare(key);
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter the plain text: ");
        String plainText = scan.nextLine();
        String preparedText = preparePlainText(plainText);
        String encryptedText = encrypt(preparedText);
        String decryptedText = decrypt(encryptedText);
        System.out.println("Key Square:");
        for (char[] row : keySquare) {
          System.out.println(Arrays.toString(row));
        }
        System.out.println("\nPlain Text: " + plainText);
        System.out.println("Prepared Text: " +
preparedText);
        System.out.println("Encrypted Text: " +
encryptedText);
        System.out.println("Decrypted Text: " +
decryptedText);
    }
}
```

Output:

```
TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\CSS Practical>  & 'C:\Program Files\Java\jdk-16.0.1\bin\java.exe' '-XX:+ShowCodeDetai
rage\54764395f21917cf820a1943ec9dad69\redhat.java\jdt_ws\CSS Practical_ac5f13fe\bin' 'P2'
Enter the plain text:
environment
Key Square:
[A, B, C, D, E]
[F, G, H, I, K]
[L, M, N, O, P]
[Q, R, S, T, U]
[V, W, X, Y, Z]

Plain Text: environment
Prepared Text: ENVIRONMENTX
Encrypted Text: CPYFTMONCPSY
Decrypted Text: ENVIRONMENTX
PS E:\CSS Practical>
```

Assignment-3

Code:

```
import java.util.*;

class RailFence {
   // function to encrypt a message
   public static String encryptRailFence(String text, int
key) {
      // create the matrix to cipher plain text
      // key = rows , length(text) = columns
      char[][] rail = new char[key][text.length()];
      // filling the rail matrix to distinguish filled
      // spaces from blank ones
      for (int i = 0; i < key; i++)
         Arrays.fill(rail[i], '\n');
      boolean dirDown = false;
      int row = 0, col = 0;
      for (int i = 0; i < text.length(); i++) {
         // check the direction of flow
         // reverse the direction if we've just
         // filled the top or bottom rail
         if (row == 0 || row == key - 1)
            dirDown = !dirDown;
         // fill the corresponding alphabet
         rail[row][col++] = text.charAt(i);
         // find the next row using direction flag
         if (dirDown)
            row++;
         else
            row--;
      }
      // now we can construct the cipher using the rail
      // matrix
      StringBuilder result = new StringBuilder();
      for (int i = 0; i < key; i++)
         for (int j = 0; j < text.length(); j++)
            if (rail[i][j] != '\n')
               result.append(rail[i][j]);
      return result.toString();
   }

   // This function receives cipher-text and key
   // and returns the original text after decryption
   public static String decryptRailFence(String cipher,
int key) {
      // create the matrix to cipher plain text
      // key = rows , length(text) = columns
      char[][] rail = new char[key][cipher.length()];
      // filling the rail matrix to distinguish filled
      // spaces from blank ones
      for (int i = 0; i < key; i++)
         Arrays.fill(rail[i], '\n');
      // to find the direction
      boolean dirDown = true;
      int row = 0, col = 0;
      // mark the places with '*'
      for (int i = 0; i < cipher.length(); i++) {
         // check the direction of flow
         if (row == 0)
            dirDown = true;
         if (row == key - 1)
            dirDown = false;
         // place the marker
         rail[row][col++] = '*';
         // find the next row using direction flag
         if (dirDown)
            row++;
         else
            row--;
      }
      // now we can construct the fill the rail matrix
      int index = 0;
      for (int i = 0; i < key; i++)
         for (int j = 0; j < cipher.length(); j++)
            if (rail[i][j] == '*'
                  && index < cipher.length())
               rail[i][j] = cipher.charAt(index++);
      StringBuilder result = new StringBuilder();
      row = 0;
      col = 0;
      for (int i = 0; i < cipher.length(); i++) {
         // check the direction of flow
         if (row == 0)
            dirDown = true;
         if (row == key - 1)
            dirDown = false;
         if (rail[row][col] != '*')
            result.append(rail[row][col++]);
         // find the next row using direction flag
         if (dirDown)
            row++;
         else
            row--;
      }
      return result.toString();
   }

   public static void main(String[] args) {
      // Encryption
      System.out.println("Encrypted Message: ");
      System.out.println(encryptRailFence("attack at
once", 2));
      System.out.println(encryptRailFence("GeeksforGe
eks ", 3));
      System.out.println(encryptRailFence("defend the
east wall", 3));
```

```
    // Now decryption of the same cipher-text
    System.out.println("\nDecrypted Message: ");
    System.out.println(decryptRailFence("atc toctaka
ne", 2));
    System.out.println(decryptRailFence("GsGsekfrek
eoe", 3));
    System.out.println(decryptRailFence("dnhaweedt
ees alf  tl", 3));
  }
}
```

Output:

```
TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improve

PS E:\CSS Practical>  & 'C:\Program Files\Java\jdk-16.0.1\
User\workspaceStorage\54764395f21917cf820a1943ec9dad69\red
Encrypted Message:
atc toctaka ne
GsGsekfrek eoe
dnhaweedtees alf  tl

Decrypted Message:
attack at once
GeeksforGeeks
defend the east wall
PS E:\CSS Practical>
```

Assignment-4

Code:

```
import java.util.*;

public class P4 {
   public static String encrypt(String message, String keyword) {
      // Create a matrix to store the plaintext message.
      int keyLength = keyword.length();
      char[][] matrix = new char[keyLength][message.length()];

      // Write the plaintext message to the matrix in a columnar fashion.
      int row = 0;
      int col = 0;
      for (int i = 0; i < message.length(); i++) {
         matrix[row][col] = message.charAt(i);
         row++;
         if (row == keyLength) {
            row = 0;
            col++;
         }
      }

      // Order the columns by the alphabetical order of the keyword.
      int[] columnOrder = new int[keyLength];
      for (int i = 0; i < keyLength; i++) {
         columnOrder[i] = i;
      }

      // Sort the column order based on the characters in the keyword.
      Arrays.sort(columnOrder, new Comparator<Integer>() {
         @Override
         public int compare(Integer o1, Integer o2) {
            return Character.compare(keyword.charAt(o1), keyword.charAt(o2));
         }
      });

      // Read the ciphertext off column by column, in the order specified by the
      // column order.
      StringBuilder ciphertext = new StringBuilder();
      for (int i = 0; i < keyLength; i++) {
         for (int j = 0; j < message.length(); j++) {
            ciphertext.append(matrix[j][columnOrder[i]]);
         }
      }

      return ciphertext.toString();
   }

   public static String decrypt(String ciphertext, String keyword) {
      // Create a matrix to store the ciphertext.
      int keyLength = keyword.length();
      char[][] matrix = new char[keyLength][ciphertext.length()];

      // Order the columns by the alphabetical order of the keyword.
```

```
    int[] columnOrder = new int[keyLength];
    for (int i = 0; i < keyLength; i++) {
      columnOrder[i] = i;
    }

    // Sort the column order based on the characters in the keyword.
    Arrays.sort(columnOrder, new Comparator<Integer>() {
      @Override
      public int compare(Integer o1, Integer o2) {
        return Character.compare(keyword.charAt(o1), keyword.charAt(o2));
      }
    });

    // Write the ciphertext to the matrix, in the order specified by the column
    // order.
    int row = 0;
    int col = 0;
    for (int i = 0; i < ciphertext.length(); i++) {
      matrix[row][columnOrder[col]] = ciphertext.charAt(i);
      col++;
      if (col == keyLength) {
        col = 0;
        row++;
      }
    }

    // Read the plaintext off row by row, from left to right.
    StringBuilder plaintext = new StringBuilder();
    for (int i = 0; i < matrix[0].length; i++) {
      for (int j = 0; j < keyLength; j++) {
        plaintext.append(matrix[j][i]);
      }
    }

    return plaintext.toString();
  }

  public static void main(String[] args) {
    String message = "SECRET MESSAGE";
    String keyword = "ZEBRAS";

    String ciphertext = encrypt(message, keyword);
    System.out.println("Ciphertext: " + ciphertext);

    String plaintext = decrypt(ciphertext, keyword);
    System.out.println("Plaintext: " + plaintext);
  }
}
```

Output:

```
Ciphertext: SECMRETESSAGE

Plaintext: SECRET MESSAGE
```

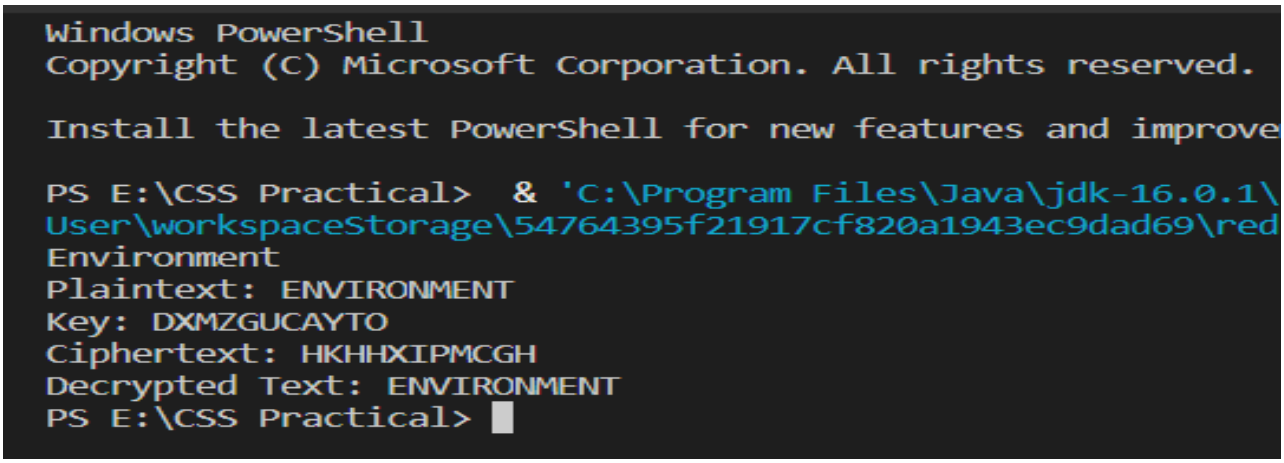
Assignment-5

Code:

```
import java.util.Random;
import java.util.Scanner;

public class P5 {
   // Function to generate a random key (pad) of the same length as the plaintext
   public static String generateRandomKey(int length) {
      Random random = new Random();
      StringBuilder keyBuilder = new StringBuilder();
      for (int i = 0; i < length; i++) {
         char randomChar = (char) (random.nextInt(26) + 'A'); // Generates a random uppercase letter
         keyBuilder.append(randomChar);
      }
      return keyBuilder.toString();
   }

   // Function to perform one-time pad encryption
   public static String encrypt(String plaintext, String key) {
      if (plaintext.length() != key.length()) {
         throw new IllegalArgumentException("Plaintext and key must have the same length.");
      }
      StringBuilder ciphertextBuilder = new StringBuilder();
      for (int i = 0; i < plaintext.length(); i++) {
         char encryptedChar = (char) ((plaintext.charAt(i) + key.charAt(i)) % 26 + 'A');
         ciphertextBuilder.append(encryptedChar);
      }
      return ciphertextBuilder.toString();
   }

   // Function to perform one-time pad decryption
   public static String decrypt(String ciphertext, String key) {
      if (ciphertext.length() != key.length()) {
         throw new IllegalArgumentException("Ciphertext and key must have the same length.");
      }
      StringBuilder decryptedBuilder = new StringBuilder();
      for (int i = 0; i < ciphertext.length(); i++) {
         char decryptedChar = (char) ((ciphertext.charAt(i) - key.charAt(i) + 26) % 26 + 'A');
         decryptedBuilder.append(decryptedChar);
      }
      return decryptedBuilder.toString();
   }

   public static void main(String[] args) {
      // Input string from the user
      Scanner scan = new Scanner(System.in);
      String randomtext = scan.nextLine();
      String plaintext = randomtext.toUpperCase();
      String key = generateRandomKey(plaintext.length());

      System.out.println("Plaintext: " + plaintext);
      System.out.println("Key: " + key);

      String ciphertext = encrypt(plaintext, key);
```

```
        System.out.println("Ciphertext: " + ciphertext);

        String decryptedText = decrypt(ciphertext, key);
        System.out.println("Decrypted Text: " + decryptedText);
    }
}
```

Output:

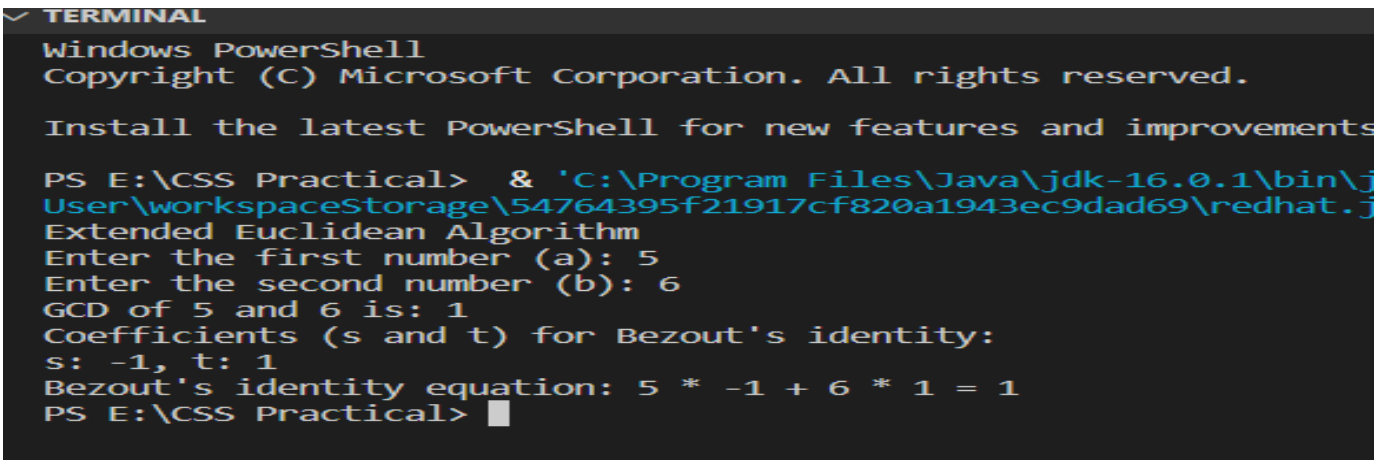


Assignment-6

Code:

```
import java.util.Scanner;

public class P6 {
   // Function to perform the extended Euclidean algorithm
   public static int[] extendedEuclidean(int a, int b) {
      if (b == 0) {
         return new int[] { a, 1, 0 };
      }
      int[] values = extendedEuclidean(b, a % b);
      int gcd = values[0];
      int s = values[2];
      int t = values[1] - (a / b) * values[2];
      return new int[] { gcd, s, t };
   }

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.println("Extended Euclidean Algorithm");
      System.out.print("Enter the first number (a): ");
      int a = scanner.nextInt();
      System.out.print("Enter the second number (b): ");
      int b = scanner.nextInt();
      scanner.close();
      int[] values = extendedEuclidean(a, b);
      int gcd = values[0];
      int s = values[1];
      int t = values[2];
      System.out.println("GCD of " + a + " and " + b + " is: " + gcd);
      System.out.println("Coefficients (s and t) for Bezout's identity:");
      System.out.println("s: " + s + ", t: " + t);
      System.out.println("Bezout's identity equation: " + a + " * " + s + " + " + b + " * " + t + " = " + gcd);
   }
}
```

Output:



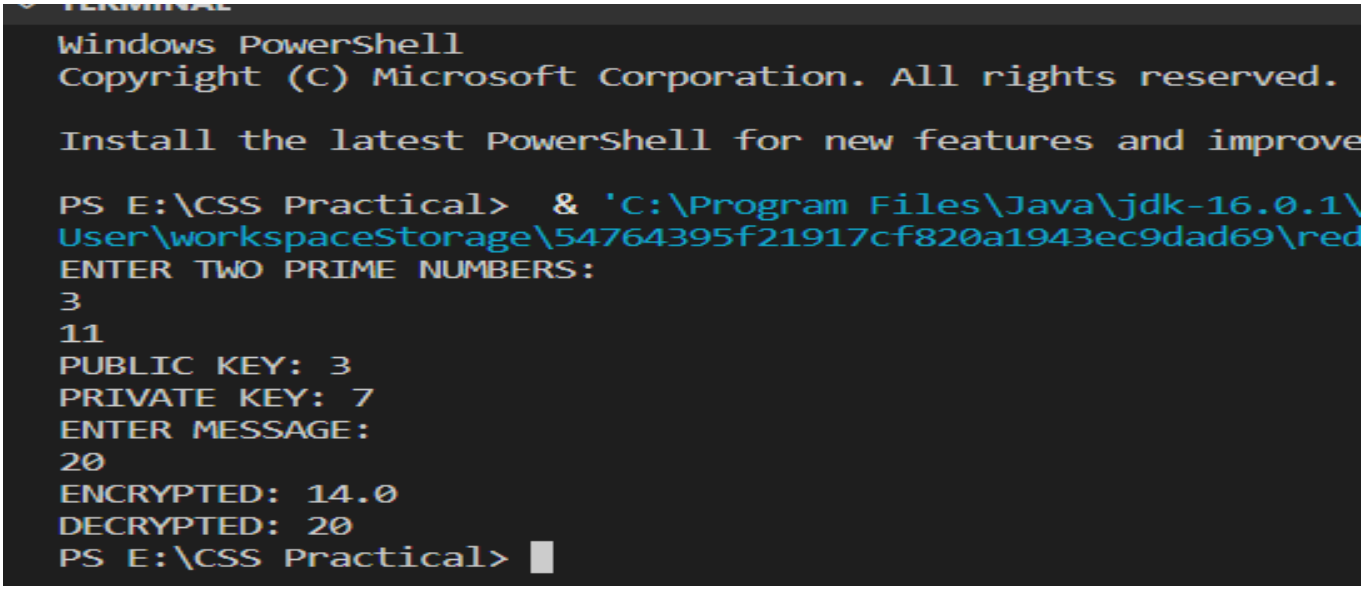
Assignment-7

Code:

```
import java.io.*;

public class P7 {
    public static void main(String args[]) throws IOException {
        int q, p, n, pn, publickey = 0, d = 0, msg;
        double cipher, ptext;

        DataInputStream in = new DataInputStream(System.in);
        System.out.println("ENTER TWO PRIME NUMBERS:");
        p = Integer.parseInt(in.readLine());
        q = Integer.parseInt(in.readLine());

        if (!isPrime(p) || !isPrime(q)) {
            System.out.println("Both numbers should be prime. Exiting...");
            System.exit(0);
        }

        n = p * q;
        pn = (p - 1) * (q - 1);

        for (int e = 2; e < pn; e++) {
            if (gcd(e, pn) == 1) {
                publickey = e;
                System.out.println("PUBLIC KEY: " + e);
                break;
            }
        }

        for (int i = 0; i < pn; i++) {
            d = i;
            if ((d * publickey % pn) == 1)
                break;
        }

        System.out.println("PRIVATE KEY: " + d);

        System.out.println("ENTER MESSAGE:");
        msg = Integer.parseInt(in.readLine());

        cipher = Math.pow(msg, publickey);
        cipher = cipher % n;
        System.out.println("ENCRYPTED: " + cipher);

        ptext = Math.pow(cipher, d);
        ptext = ptext % n;
        System.out.println("DECRYPTED: " + (int) ptext);
    }

    static boolean isPrime(int a) {
        if (a <= 1)
            return false;
        for (int i = 2; i * i <= a; i++) {
            if (a % i == 0) {
                return false;
            }
        }
        return true;
    }

    static int gcd(int number1, int number2) {
        if (number2 == 0) {
            return number1;
        }
```

```
        return gcd(number2, number1 % number2);
    }
}
```

Output:



Assignment-8

Code:

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

public class P8 {
   public static void main(String[] args) throws Exception {
      // Generate a key pair
      KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");
      keyPairGenerator.initialize(2048);
      KeyPair keyPair = keyPairGenerator.generateKeyPair();
      // Get the private key
      PrivateKey privateKey = keyPair.getPrivate();
      // Get the message to be signed
      String message = "This is a message to be signed.";
      // Create a signature object
      Signature signature = Signature.getInstance("SHA256withRSA");
      // Initialize the signature object with the private key
      signature.initSign(privateKey);
      // Add the message to the signature object
      signature.update(message.getBytes());
      // Calculate the signature
      byte[] signatureBytes = signature.sign();
      // Save the signature
      String signatureString = Base64.getEncoder().encodeToString(signatureBytes);
      System.out.println("Signature: " + signatureString);
      // Verify the signature
      Signature verificationSignature = Signature.getInstance("SHA256withRSA");
      // Initialize the verification signature object with the public key
      verificationSignature.initVerify(keyPair.getPublic());
      // Add the message to the verification signature object
      verificationSignature.update(message.getBytes());
      // Verify the signature
      boolean isVerified = verificationSignature.verify(signatureBytes);
      System.out.println("Signature verified: " + isVerified);
   }
```

Output:


TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS E:\CSS Practical> & 'C:\Program Files\Java\jdk-16.0.1\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\91860\AppData\Roaming\Code\User\workspaceStorage\54764395f21917cf820a1943ec9dad69\redhat.java\jdt_ws\CSS Practical_ac5f13fe\bin' 'P8'
Signature: NO5LTnqqmCdfIgiB6fq4192NWwRswq/oDzVthH2P3EKvV+TrLEbm9R3cxs5DWzU6O6dBk+St1QZs5eYKTnXftpCIv6jfunwpW4UflQw/euotnUEsg6VjsO+yCMGJjaTWH9WEmGiNrHyunVjqcBO9wtf97GrpjS3kFH7p6oPl1SCoNxZ08s0MgMLf2/25J2XyLxyQBxBT+RJ5dGmpMQnAoGZWMUewbJ5yoeKzaiWaOTq61GEqdXdjSBJfrTLVfacTb5e1q0oO9/CylPIiEV+kyZl+wxAbhYq73b5w/gayMdupOQaViSDuxlRaDMPOJH/ZtCTBNcbD8rysMMKbW8Zesg==
Signature verified: true
PS E:\CSS Practical>
```
```