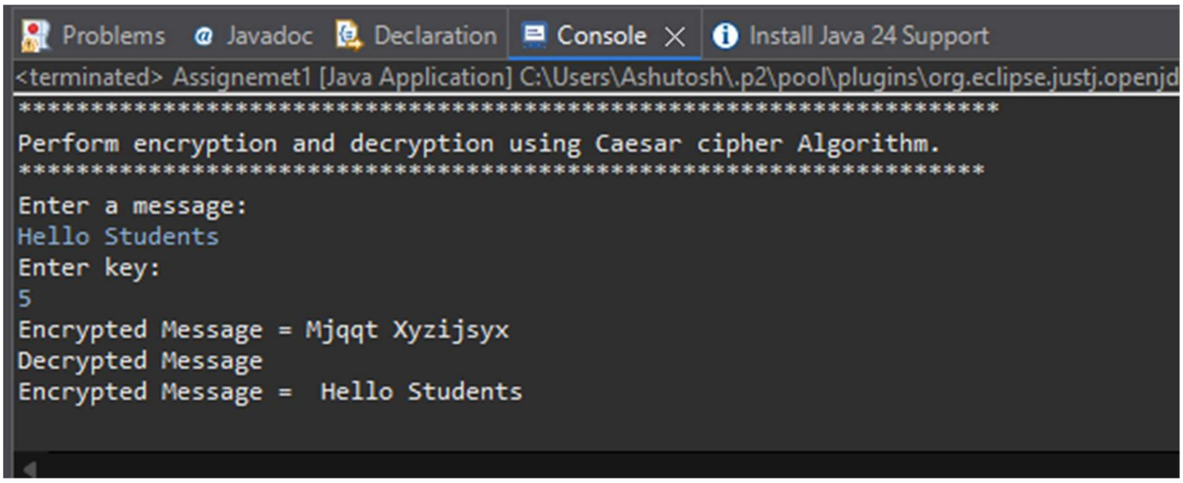
Assignment No 1

Code:

```
import java.util.Scanner;

public class Assignemet1 {
	public static void main(String... s) {
		String message, encryptedMessage = "", decryptedMessage = " ";
		int key;
		char ch;
		Scanner sc = new Scanner(System.in);

	System.out.println("*****************************************************");
	System.out.println("Perform encryption and decryption using Caesar cipher 
Algorithm.  ");

	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);
		System.out.println("Decrypted Message");
		for (int i = 0; i < encryptedMessage.length(); ++i) {
			ch = encryptedMessage.charAt(i);
			if (ch >= 'a' && ch <= 'z') {
```

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

Output :



```
<terminated> Assignemet1 [Java Application] C:\Users\Ashutosh\.p2\pool\plugins\org.eclipse.justj.openjd
*******************************************************************
Perform encryption and decryption using Caesar cipher Algorithm.
******************************************************************
Enter a message:
Hello Students
Enter key:
5
Encrypted Message = Mjqqt Xyzijsyx
Decrypted Message
Encrypted Message =  Hello Students
```

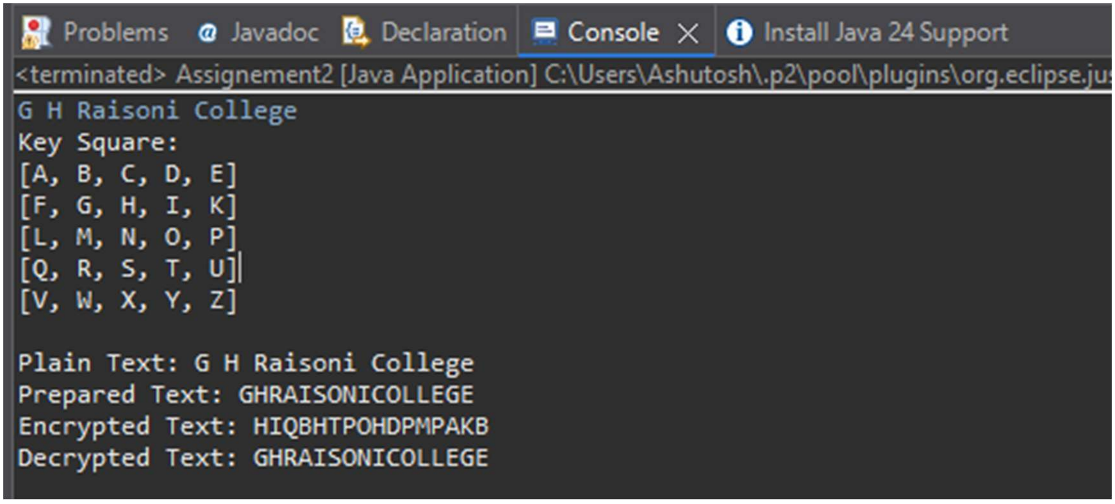
Assignment No 2

Code:

```
import java.util.Arrays;
import java.util.Scanner;
public class Assignement2 {
    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); // Take input from user using scanner
class
        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:



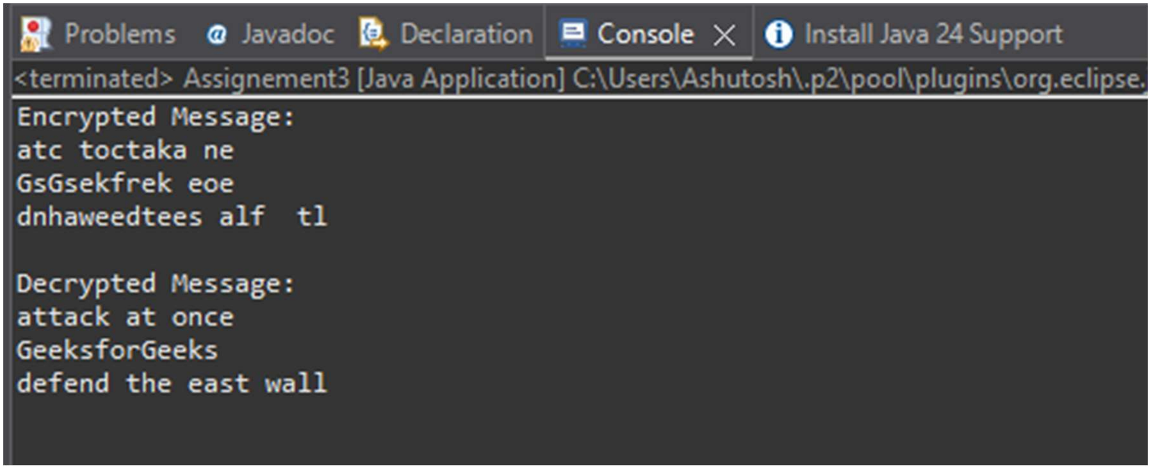
```
<terminated> Assignement2 [Java Application] C:\Users\Ashutosh\.p2\pool\plugins\org.eclipse.ju
G H Raisoni College
Key Square:
[A, B, C, D, E]
[F, G, H, I, K]
[L, M, N, O, P]
[Q, R, S, T, U]
[V, W, X, Y, Z]

Plain Text: G H Raisoni College
Prepared Text: GHRAISONICOLLEGE
Encrypted Text: HIQBHTPOHDPMPAKB
Decrypted Text: GHRAISONICOLLEGE
```

Assignment No 3

Code:

```
import java.util.Arrays;

class Assignement3 {
    public static String encryptRailFence(String text, int key) {
        char[][] rail = new char[key][text.length()];
        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++) {
            if (row == 0 || row == key - 1)
                dirDown = !dirDown;
            rail[row][col++] = text.charAt(i);
            if (dirDown)
                row++;
            else
                row--;
        }
        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();
    }

    public static String decryptRailFence(String cipher, int key) {
        char[][] rail = new char[key][cipher.length()];
        for (int i = 0; i < key; i++)
            Arrays.fill(rail[i], '\n');
        boolean dirDown = true;
        int row = 0, col = 0;
        for (int i = 0; i < cipher.length(); i++) {
            if (row == 0)
                dirDown = true;
            if (row == key - 1)
                dirDown = false;
            rail[row][col++] = '*';
            if (dirDown)
                row++;
            else
                row--;
        }
        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++) {
            if (row == 0)
                dirDown = true;
            if (row == key - 1)
                dirDown = false;
            if (rail[row][col] != '*')
                result.append(rail[row][col++]);
            if (dirDown)
                row++;
            else
                row--;
        }
        return result.toString();
    }

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

        System.out.println("\nDecrypted Message: ");
        System.out.println(decryptRailFence("atc toctaka ne", 2));
        System.out.println(decryptRailFence("GsGsekfrek eoe", 3));
        System.out.println(decryptRailFence("dnhaweedtees alf  tl", 3));
    }
}
```

Output:



<terminated> Assignement3 [Java Application] C:\Users\Ashutosh\.p2\pool\plugins\org.eclipse.

```
Encrypted Message:
atc toctaka ne
GsGsekfrek eoe
dnhaweedtees alf  tl

Decrypted Message:
attack at once
GeeksforGeeks
defend the east wall
```

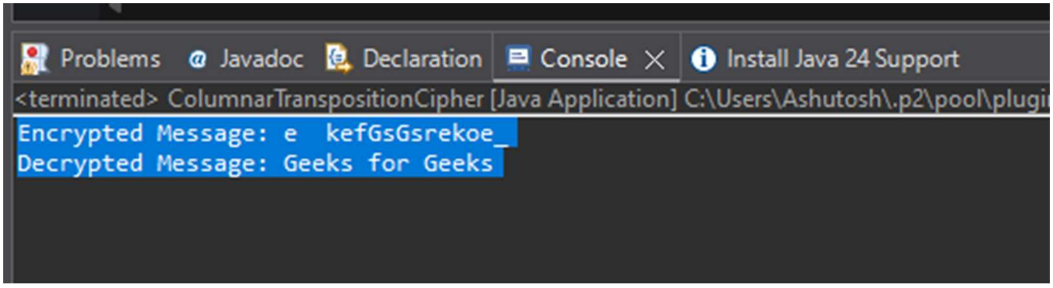
Assignment No 4

Code:

```
import java.util.*;

public class ColumnarTranspositionCipher {
    // Key for Columnar Transposition
    static final String key = "HACK";
    static Map<Character, Integer> keyMap = new HashMap<>();

    static void setPermutationOrder() {
        // Add the permutation order into the map
        for (int i = 0; i < key.length(); i++) {
            keyMap.put(key.charAt(i), i);
        }
    }

    // Encryption
    static String encryptMessage(String msg) {
        int row, col;
        StringBuilder cipher = new StringBuilder();

        /* Calculate the number of columns in the matrix */
        col = key.length();

        /* Calculate the maximum number of rows in the matrix */
        row = (int) Math.ceil((double) msg.length() / col);

        char[][] matrix = new char[row][col];

        for (int i = 0, k = 0; i < row; i++) {
            for (int j = 0; j < col; ) {
                if (k < msg.length()) {
                    char ch = msg.charAt(k);
                    if (Character.isLetter(ch) || ch == ' ') {
                        matrix[i][j] = ch;
                        j++;
                    }
                    k++;
                } else {
                    /* Add padding character '_' */
                    matrix[i][j] = '_';
                    j++;
                }
            }
        }
    for (Map.Entry<Character, Integer> entry : keyMap.entrySet()) {
      int columnIndex = entry.getValue();

      // Get the cipher text from the matrix column-wise using the permuted key
      for (int i = 0; i < row; i++) {
        if (Character.isLetter(matrix[i][columnIndex]) || matrix[i][columnIndex] == ' ' ||
matrix[i][columnIndex] == '_') {
          cipher.append(matrix[i][columnIndex]);
        }
      }
    }

    return cipher.toString();
  }

  // Decryption
  static String decryptMessage(String cipher) {
    /* Calculate the number of columns for the cipher matrix */
    int col = key.length();

    int row = (int) Math.ceil((double) cipher.length() / col);
    char[][] cipherMat = new char[row][col];

    /* Add characters into the matrix column-wise */
    int k = 0;
    for (int j = 0; j < col; j++) {
      for (int i = 0; i < row; i++) {
        cipherMat[i][j] = cipher.charAt(k);
        k++;
      }
    }

    /* Update the order of the key for decryption */
    int index = 0;
    for (Map.Entry<Character, Integer> entry : keyMap.entrySet()) {
      entry.setValue(index++);
    }

    /* Arrange the matrix column-wise according to the permutation order */
    char[][] decCipher = new char[row][col];
    for (int l = 0; l < key.length(); l++) {
      int columnIndex = keyMap.get(key.charAt(l));
      for (int i = 0; i < row; i++) {
        decCipher[i][l] = cipherMat[i][columnIndex];
      }
    }
```

```
        /* Get the message using the matrix */
        StringBuilder msg = new StringBuilder();
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                if (decCipher[i][j] != '_') {
                    msg.append(decCipher[i][j]);
                }
            }
        }

        return msg.toString();
    }

    public static void main(String[] args) {
        /* Message */
        String msg = "Geeks for Geeks";

        setPermutationOrder();

        // Calling encryption function
        String cipher = encryptMessage(msg);
        System.out.println("Encrypted Message: " + cipher);

        // Calling Decryption function
        System.out.println("Decrypted Message: " + decryptMessage(cipher));
    }
}
```

Output:



```
<terminated> ColumnarTranspositionCipher [Java Application] C:\Users\Ashutosh\.p2\pool\plugi
Encrypted Message: e  kefGsGsrekoe_
Decrypted Message: Geeks for Geeks
```

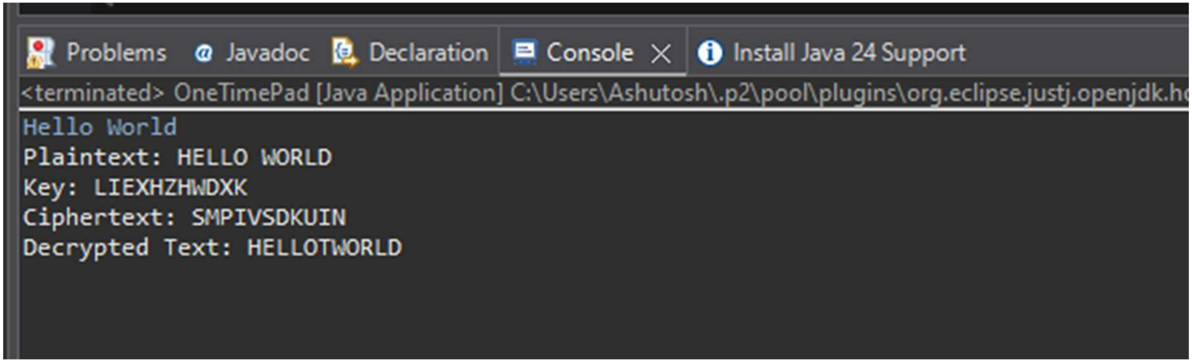
Assignment No 5

Code:

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

public class OneTimePad {
// 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 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:



```
<terminated> OneTimePad [Java Application] C:\Users\Ashutosh\.p2\pool\plugins\org.eclipse.justj.openjdk.h
Hello World
Plaintext: HELLO WORLD
Key: LIEXHZHWDXK
Ciphertext: SMPIVSDKUIN
Decrypted Text: HELLOTWORLD
```

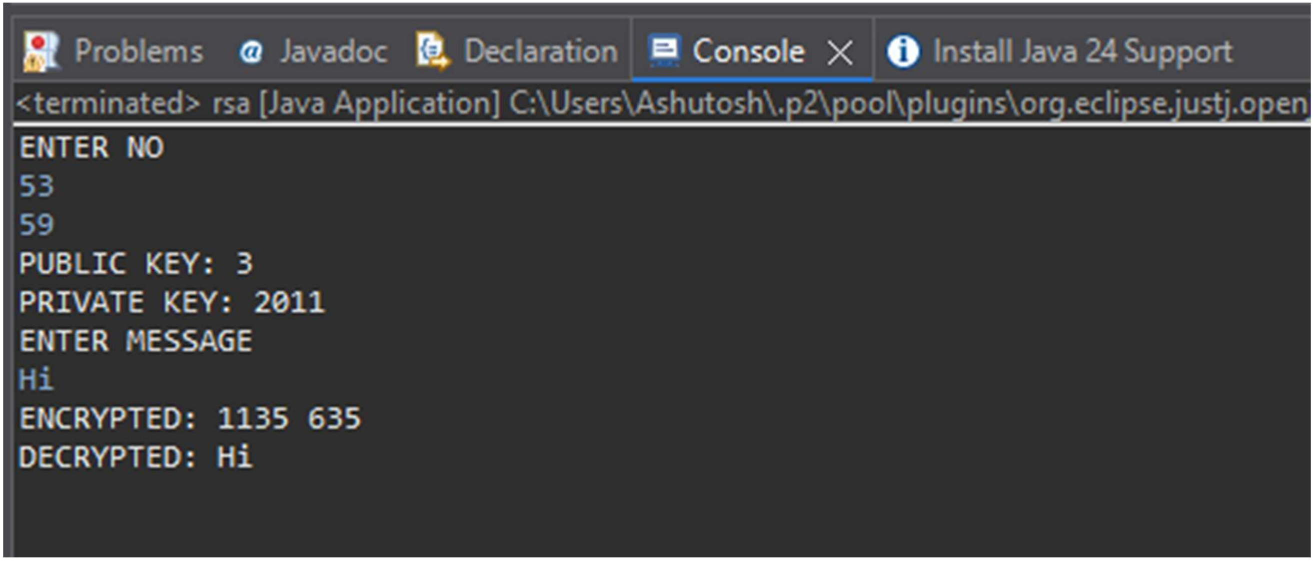
Assignment No 6

Code:

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

class rsa {
   public static void main(String args[]) throws IOException {
      int q, p, n, pn, publickey = 0, d = 0, msg;
      BigInteger cipher, ptext;
      int check, check1;

      // Reading input values
      DataInputStream in = new DataInputStream(System.in);
      System.out.println("ENTER NO");
      p = Integer.parseInt(in.readLine());  // Read first prime number
      q = Integer.parseInt(in.readLine());  // Read second prime number

      check = prime(p);  // Check if p is prime
      check1 = prime(q); // Check if q is prime

      if (check != 1 || check1 != 1) {
         System.exit(0);  // Exit if p or q is not prime
      }

      n = p * q;  // Compute n
      pn = (p - 1) * (q - 1);  // Compute Euler's Totient Function

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

      // Find the private key (d)
      for (int i = 0; i < pn; i++) {
         d = i;
         if (((d * publickey) % pn) == 1) {
            break;
         }
      }
      System.out.println("PRIVATE KEY: " + d);  // Print the private key

      System.out.println("ENTER MESSAGE");
```

```
        String message = in.readLine();  // Read the message as a string

        StringBuilder cipherText = new StringBuilder();
        for (int i = 0; i < message.length(); i++) {
            msg = message.charAt(i);  // Get the ASCII value of each character in the message
            cipher = new BigInteger(String.valueOf(msg)).modPow(new
BigInteger(String.valueOf(publickey)), new BigInteger(String.valueOf(n)));  // Encrypt using
RSA formula
            cipherText.append(cipher.toString()).append(" ");  // Store encrypted value (as a
string)
        }

        System.out.println("ENCRYPTED: " + cipherText.toString());  // Print the encrypted
message

        // Decrypt the ciphertext
        String[] encryptedValues = cipherText.toString().split(" ");
        StringBuilder decryptedMessage = new StringBuilder();
        for (String encryptedValue : encryptedValues) {
            BigInteger encryptedNum = new BigInteger(encryptedValue);  // Get the encrypted
number
            ptext = encryptedNum.modPow(new BigInteger(String.valueOf(d)), new
BigInteger(String.valueOf(n)));  // Decrypt using RSA formula
            decryptedMessage.append((char)ptext.intValue());  // Convert the decrypted number
back to a character
        }

        System.out.println("DECRYPTED: " + decryptedMessage.toString());  // Print the
decrypted message
    }

    // Method to check if a number is prime
    static int prime(int a) {
        int flag = 0;
        for (int i = 2; i < a; i++) {
            if (a % i == 0) {
                System.out.println(a + " is not a Prime Number");
                flag = 1;
                return 0;  // Return 0 if not prime
            }
        }
        if (flag == 0)
            return 1;  // Return 1 if prime
        return 1;
    }
    // Method to compute the GCD of two numbers
    static int gcd(int number1, int number2) {
        if (number2 == 0) {
            return number1;
        }
        return gcd(number2, number1 % number2);
    }
}
```

Output:



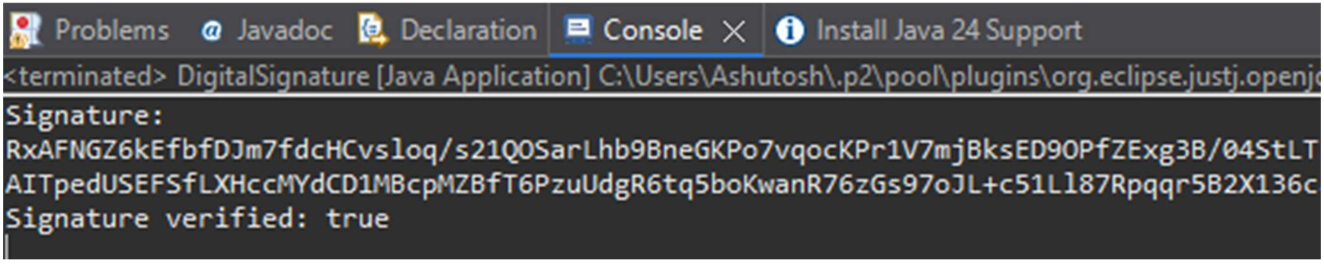
```
<terminated> rsa [Java Application] C:\Users\Ashutosh\.p2\pool\plugins\org.eclipse.justj.open
ENTER NO
53
59
PUBLIC KEY: 3
PRIVATE KEY: 2011
ENTER MESSAGE
Hi
ENCRYPTED: 1135 635
DECRYPTED: Hi
```

Assignment No 7

Code:

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

public class DigitalSignature {
        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:



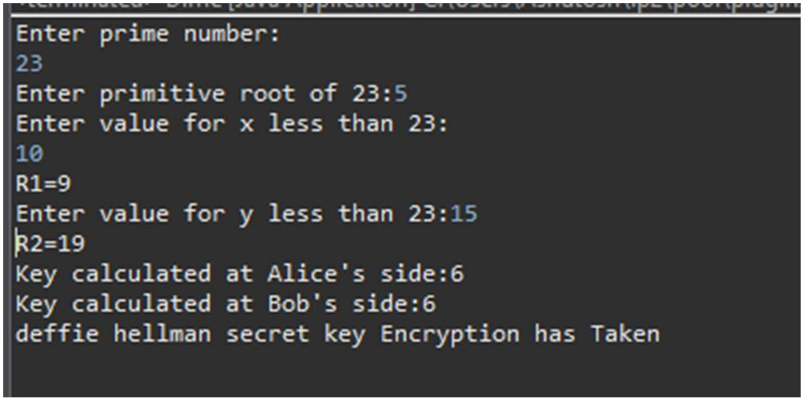
```
<terminated> DigitalSignature [Java Application] C:\Users\Ashutosh\.p2\pool\plugins\org.eclipse.justj.openj
Signature:
RxAFNGZ6kEfbfDJm7fdcHCvsloq/s21QOSarLhb9BneGKPo7vqocKPr1V7mjBksED9OPfZExg3B/04StLT
AITpedUSEFSfLXHccMYdCD1MBcpMZBfT6PzuUdgR6tq5boKwanR76zGs97oJL+c51Ll87Rpqqr5B2X136c
Signature verified: true
```

Assignment No 8

Code:

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

class Diffie {
        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("Key calculated at Alice's side:" + k1);
                BigInteger k2 = R1.modPow(y, p);
                System.out.println("Key calculated at Bob's side:" + k2);
                System.out.println("deffie hellman secret key Encryption has Taken");
        }
}
```

Output:



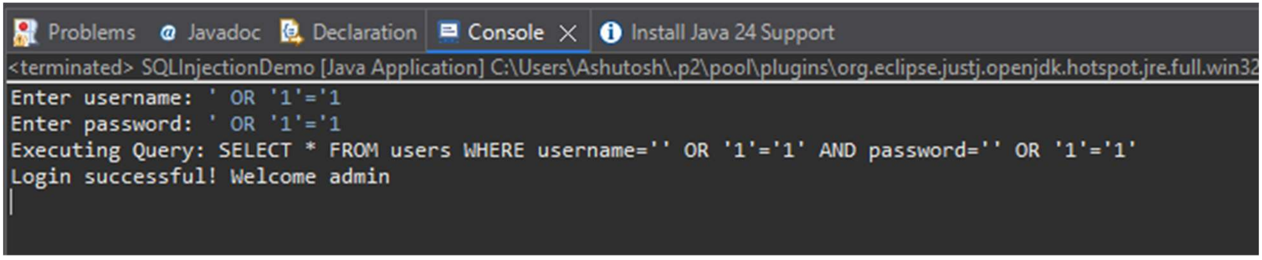
```
Enter prime number:
23
Enter primitive root of 23:5
Enter value for x less than 23:
10
R1=9
Enter value for y less than 23:15
R2=19
Key calculated at Alice's side:6
Key calculated at Bob's side:6
deffie hellman secret key Encryption has Taken
```

Assignment No 9

Code:

```
import java.sql.*;
import java.util.Scanner;

public class SQLInjectionDemo {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/demo"; // Change DB name, port as needed
        String user = "root"; // Replace with your DB username
        String password = "Ashu@13"; // Replace with your DB password
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter username: ");
        String inputUsername = scanner.nextLine();
        System.out.print("Enter password: ");
        String inputPassword = scanner.nextLine();
        try {
            Connection conn = DriverManager.getConnection(url, user, password);
            Statement stmt = conn.createStatement();
            String query = "SELECT * FROM users WHERE username='" + inputUsername + "'
AND password='" + inputPassword + "'";
            System.out.println("Executing Query: " + query);
            ResultSet rs = stmt.executeQuery(query);
            if (rs.next()) {
                System.out.println("Login successful! Welcome " + rs.getString("username"));
            } else {
                System.out.println("Invalid credentials.");
            }
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        scanner.close();
    }
}
```

Output:



```
<terminated> SQLInjectionDemo [Java Application] C:\Users\Ashutosh\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32
Enter username: ' OR '1'='1
Enter password: ' OR '1'='1
Executing Query: SELECT * FROM users WHERE username='' OR '1'='1' AND password='' OR '1'='1'
Login successful! Welcome admin
```