ADITYA

COLLEGE OF ENGINEERING & TECHNOLOGY

An AUTONOMOUS Institution

Approved by AICTE, Permanently Affiliated to JNTUK,
Accredited by NBA & NAAC with A+ Grade
Recognized by UGC under Section 2(f) and 12(B) of UGC Act, 1956
Aditya Nagar, ADB Road, Surampalem, Kakinada District - 533437, A.P.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LABORATORY RECORD

NAME :

ROLL NO :

YEAR :

SEMESTER :

LAB :

ADITYA

COLLEGE OF ENGINEERING & TECHNOLOGY

An AUTONOMOUS Institution

Approved by AICTE, Permanently Affiliated to JNTUK,
Accredited by NBA & NAAC with A+ Grade
Recognized by UGC under Section 2(f) and 12(B) of UGC Act, 1956
Aditya Nagar, ADB Road, Surampalem, Kakinada District - 533437, A.P.

Department of
COMPUTER SCIENCE AND ENGINEERING

Name : **Roll No. :** □□□□□□□□□□

Certified that this is the bonafide record of practical work done by

Mr. /Ms. ....

a student of ....with PIN No. ....

in the .... Laboratory during the year ....

No. of Practicals Conducted : □ No. of Practicals Attended : □

Signature - Faculty Incharge **Signature - Head of the Department**

Submitted for the Practical examination held on ....

EXAMINER - 1 **EXAMINER - 2**

ADITYA COLLEGE OF ENGINEERING AND TECHNOLOGY

INSTITUTE VISION AND MISSION

VISION:

To induce higher planes of learning by imparting technical education with

- ✓ International standards
- ✓ Applied research
- ✓ Creative Ability
- ✓ Value based instruction and to emerge as a premiere institute

MISSION:

Achieving academic excellence by providing globally acceptable technical education by forecasting technology through

- ✓ Innovative Research And development
- ✓ Industry Institute Interaction
- ✓ Empowered Manpower

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT VISION AND MISSION

VISION:

To become a center for excellence in Computer Science and Engineering education and innovation.

MISSION:

- Provide state of art infrastructure
- Adapt skill-based learner centric teaching methodology
- Organize socio cultural events for better society
- Undertake collaborative works with academia and industry
- Encourage students and staff self-motivated, problem-solving individuals using Artificial Intelligence
- Encourage entrepreneurship in young minds.

Pointer

S No	Date	Name of the Experiment	Page No	Remark

Pointer

S No	Date	Name of the Experiment	Page No	Remark

Experiment-1

Write a C program that contains a string(char pointer) with a value\Hello World'. The programs should XOR each character in this string with 0 and display the result.

```
#include<stdlib.h>
main()
{
char str[]="Hello World";

char str1[11];
int i,len;
len=strlen(str);
for(i=0;i<len;i++)
{
str1[i]=str[i]^0; printf("%c",str1[i]);
}
printf("\n");
}
```

Output:

```
HELLO WORLD
```

Experiment-2

Write a C program that contains a string (char pointer) with a value \Hello World'. The program should AND or and XOR each character in this string with 127 and display the result.

```
#include <stdio.h>
#include<stdlib.h>
void main()
{
char str[]="Hello World";
char str1[11];
char str2[11];
int i,len;
len = strlen(str);
for(i=0;i<len;i++)
{
str1[i] = str[i]&127;
printf("%c",str1[i]);
}
printf("\n");
for(i=0;i<len;i++)
{
str2[i]=str2[i]^127;
printf("%c",str2[i]);
}
printf("\n");
}
```

OUTPUT:

```
Hello World
ÇÇÇÇÇÇÇÇU
```

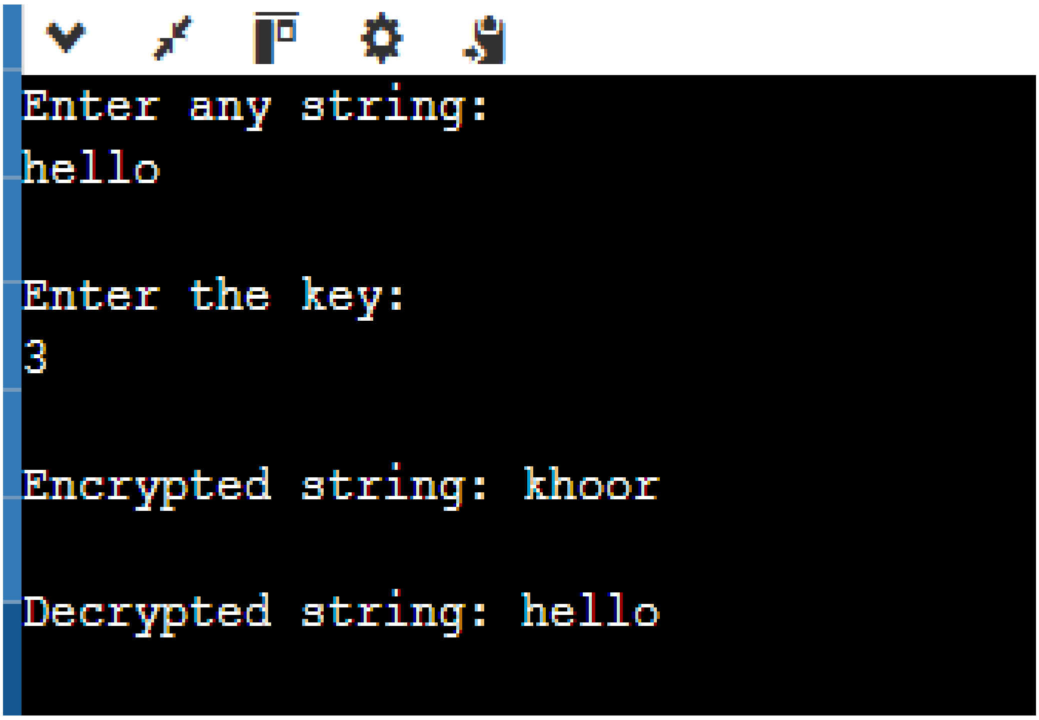
Experiment-3

Write a Java program to perform encryption and decryption using the following algorithms: i. Ceaser Cipher ii. Substitution Cipher iii. Hill Cipher

Ceaser Cipher:

```
import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.Scanner;


public class CeaserCipher {

    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.println("Enter any string:");

        String str = br.readLine();


        System.out.println("\nEnter the key:");

        int key = sc.nextInt();


        String encrypted = encrypt(str, key);

        System.out.println("\nEncrypted string: " + encrypted);


        String decrypted = decrypt(encrypted, key);

        System.out.println("\nDecrypted string: " + decrypted);

    }


    public static String encrypt(String str, int key) {

        StringBuilder encrypted = new StringBuilder();


        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.append((char) c);
        }
        return encrypted.toString();
    }
    public static String decrypt(String str, int key) {
        StringBuilder decrypted = new StringBuilder();

        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.append((char) c);
        }
        return decrypted.toString();
    }
}
```

Output:

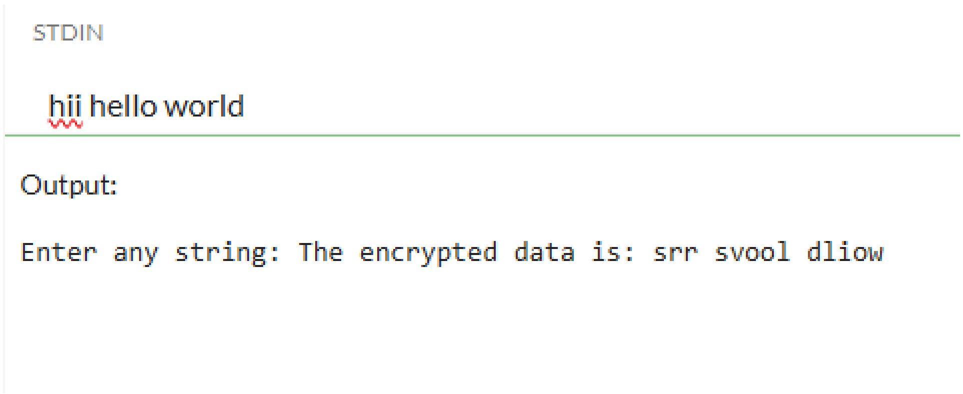


```
Enter any string:
hello

Enter the key:
3

Encrypted string: khoor

Decrypted string: hello
```

Substitution Cipher:

```
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 void main(String[] args) throws IOException {
        // Alphabet for substitution
        String a = "abcdefghijklmnopqrstuvwxyz";
        String b = "zyxwvutsrqponmlkjihgfedcba";

        // Input string from user
        System.out.print("Enter any string: ");
        String str = br.readLine().toLowerCase(); // Convert input to lowercase for consistent mapping
        String decrypt = "";

        // Encrypt the string
        for (int i = 0; i < str.length(); i++) {
            char c = str.charAt(i);
            if (Character.isLetter(c)) { // Check if the character is a letter
                int j = a.indexOf(c);
                decrypt += b.charAt(j);
            } else {
                decrypt += c; // Keep non-alphabet characters as is
            }
        }
        System.out.println("The encrypted data is: " + decrypt);
    }
}
```

Output:



STDIN

hii hello world

Output:

```
Enter any string: The encrypted data is: srr svool dliow
```

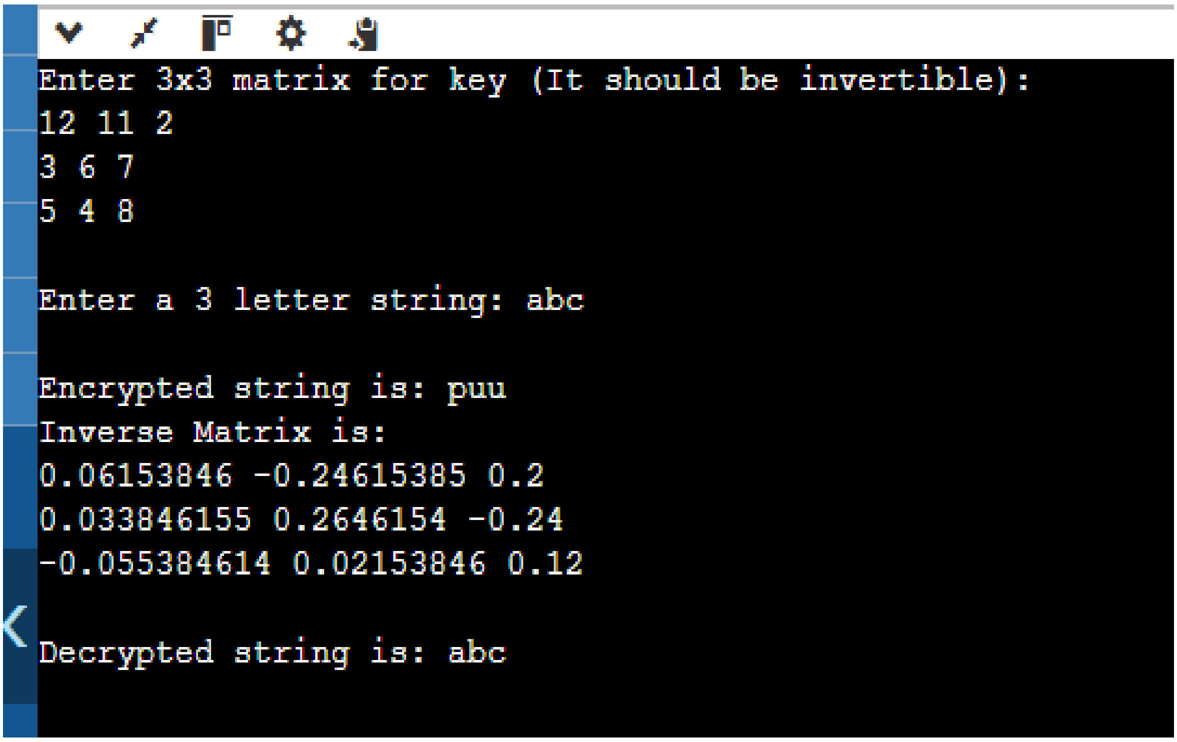
Hill Cipher:

```
import java.io.*;
import java.util.*;

public class HillCipher {

    static float[][] decrypt = new float[3][1];
    static float[][] a = new float[3][3];   // Encryption key matrix
    static float[][] b = new float[3][3];   // Inverse of key matrix
    static float[][] mes = new float[3][1]; // Message matrix
    static float[][] res = new float[3][1]; // Result matrix
    static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    static Scanner sc = new Scanner(System.in);

    public static void main(String[] args) throws IOException {
        getKeyMessage(); // Get key and message

        // Encryption process: Multiply key matrix with message matrix
        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++) {
            // Print encrypted message as characters (mod 26 to stay within alphabet)
            System.out.print((char) (Math.round(res[i][0]) % 26 + 97));
        }
        inverse(); // Calculate the inverse of the key matrix
        // Decryption process: Multiply inverse matrix with encrypted message
        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++) {
            // Print decrypted message as characters (mod 26)
            System.out.print((char) (Math.round(decrypt[i][0]) % 26 + 97));
        }
        System.out.println("\n");
    }
    // Function to get the key matrix and message
    public static void getKeyMessage() throws IOException {
        System.out.println("Enter 3x3 matrix for key (It should be invertible): ");
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                a[i][j] = sc.nextFloat(); // Enter the key matrix
            }
        }
        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; // Convert message characters to numbers (0-25)
        }
    }
    // Function to calculate the inverse of the key matrix
    public static void inverse() {
        float p, q;
```

```
float[][] c = new float[3][3];
// Copy original matrix a to matrix c
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        c[i][j] = a[i][j];
        if (i == j) {
            b[i][j] = 1; // Identity matrix for inverse
        } else {
            b[i][j] = 0;
        }
    }
}
// Gaussian elimination to calculate inverse
for (int k = 0; k < 3; k++) {
    for (int i = 0; i < 3; i++) {
        p = c[i][k];
        q = c[k][k];
        for (int j = 0; j < 3; j++) {
            if (i != k) {
                c[i][j] = c[i][j] * q - p * c[k][j];
                b[i][j] = b[i][j] * q - p * b[k][j];
            }
        }
    }
}
// Normalize the inverse matrix
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        b[i][j] = b[i][j] / c[i][i];
    }
}
```

```
        // Print the inverse matrix
        System.out.println("\nInverse Matrix is: ");
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                System.out.print(b[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Output:



```
Enter 3x3 matrix for key (It should be invertible):
12 11 2
3 6 7
5 4 8

Enter a 3 letter string: abc

Encrypted string is: puu
Inverse Matrix is:
0.06153846 -0.24615385 0.2
0.033846155 0.2646154 -0.24
-0.055384614 0.02153846 0.12

Decrypted string is: abc
```

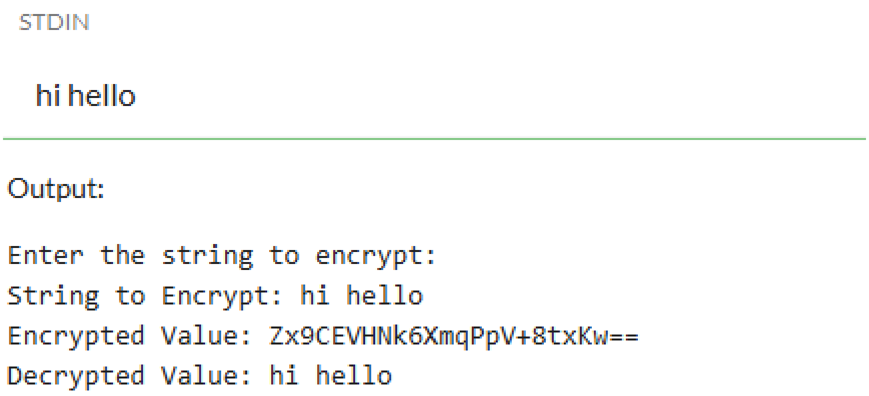
Experiment-4

Write a Java program to implement the DES algorithm logic.

```
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;
    private byte[] keyAsBytes;
    private String myEncryptionKey;
    private String myEncryptionScheme;
    private SecretKey key;
    static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    public DES() throws Exception {
        myEncryptionKey = "ThisIsASecretEncryptionKey"; // Must be at least 24 bytes
        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) {
        String encryptedString = null;
        try {
            cipher.init(Cipher.ENCRYPT_MODE, key);
            byte[] plainText = unencryptedString.getBytes(UNICODE_FORMAT);
            byte[] encryptedText = cipher.doFinal(plainText);
            encryptedString = Base64.getEncoder().encodeToString(encryptedText);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return encryptedString;
    }

    public String decrypt(String encryptedString) {
        String decryptedText = null;
        try {
            cipher.init(Cipher.DECRYPT_MODE, key);
            byte[] encryptedText = Base64.getDecoder().decode(encryptedString);
            byte[] plainText = cipher.doFinal(encryptedText);
            decryptedText = new String(plainText, UNICODE_FORMAT);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return decryptedText;
    }
    public static void main(String args[]) throws Exception {
        System.out.print("Enter the string to encrypt: ");
        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("Encrypted Value: " + encrypted);
        System.out.println("Decrypted Value: " + decrypted);
    }
```

```
}
```

Output:



STDIN

```
hi hello
```

Output:

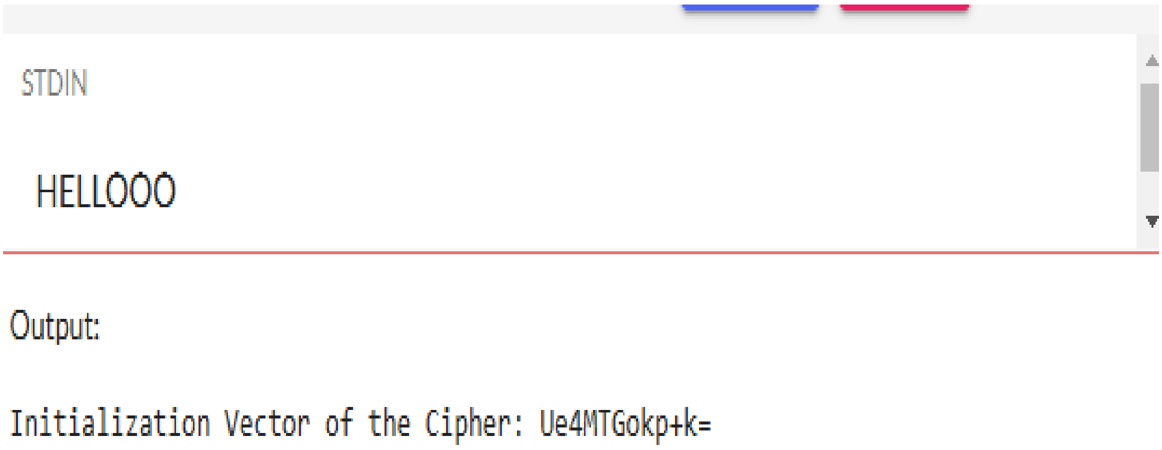
```
Enter the string to encrypt:
String to Encrypt: hi hello
Encrypted Value: Zx9CEVHNk6XmqPpV+8txKw==
Decrypted Value: hi hello
```

Experiment-5

Write a C/JAVA program to implement the BlowFish algorithm logic.

```
import java.io.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import java.security.Key;
import java.util.Base64;

public class BlowFish {
    public static void main(String[] args) throws Exception {
        // Generate the secret key for Blowfish
        KeyGenerator keyGenerator = KeyGenerator.getInstance("Blowfish");
        keyGenerator.init(128); // Blowfish key size (128 bits)
        Key secretKey = keyGenerator.generateKey();

        // Create the cipher for Blowfish in CFB mode with NoPadding
        Cipher cipherOut = Cipher.getInstance("Blowfish/CFB/NoPadding");

        // Initialize cipher for encryption
        cipherOut.init(Cipher.ENCRYPT_MODE, secretKey);

        // Get the initialization vector (IV)
        byte[] iv = cipherOut.getIV();
        if (iv != null) {
            System.out.println("Initialization Vector of the Cipher: " +
Base64.getEncoder().encodeToString(iv));
        }

        // Create file input/output streams
        FileInputStream fin = new FileInputStream("inputFile.txt");
        FileOutputStream fout = new FileOutputStream("outputFile.txt");

        // Create CipherOutputStream to encrypt while writing to the file
        CipherOutputStream cout = new CipherOutputStream(fout, cipherOut);
        int input;
        // Read bytes from input file and write to encrypted output file
        while ((input = fin.read()) != -1) {
            cout.write(input);
        }

        // Close all streams
        fin.close();
        cout.close();
        fout.close();
    }
}
```

Output:



```
STDIN

HELLOOO

Output:

Initialization Vector of the Cipher: Ue4MTGokp+k=
```

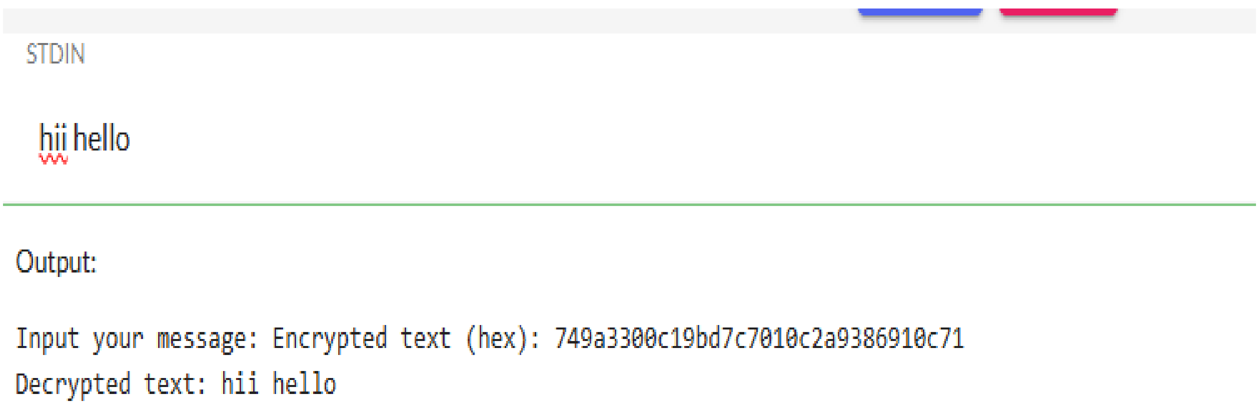
Experiment-6

Write a C/JAVA program to implement the Rijndael algorithm logic.

```
import java.security.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import java.io.*;
import java.util.Scanner;

public class AES {

  // Convert byte array to hex string
  public static String asHex(byte buf[]) {
    StringBuffer strbuf = new StringBuffer(buf.length * 2);
    int i;
    for (i = 0; i < buf.length; i++) {
      if (((int) buf[i] & 0xff) < 0x10)
        strbuf.append("0");
      strbuf.append(Long.toString((int) buf[i] & 0xff, 16));
    }
    return strbuf.toString();
  }
  public static void main(String[] args) throws Exception {

    // Create Scanner for user input
    Scanner scanner = new Scanner(System.in);

    // Prompt for user input message
    System.out.print("Input your message: ");
    String message = scanner.nextLine();

    // Get the KeyGenerator for AES
```

```
        KeyGenerator kgen = KeyGenerator.getInstance("AES");
        kgen.init(128); // AES key size (128 bits)

        // Generate the secret key
        SecretKey skey = kgen.generateKey();
        byte[] raw = skey.getEncoded();

        // Create SecretKeySpec from the raw key
        SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");

        // Instantiate the Cipher for AES/ECB/PKCS5Padding mode
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");

        // Encrypt the message
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
        byte[] encrypted = cipher.doFinal(message.getBytes());

        // Print the encrypted string in hex
        System.out.println("Encrypted text (hex): " + asHex(encrypted));

        // Decrypt the message
        cipher.init(Cipher.DECRYPT_MODE, skeySpec);
        byte[] original = cipher.doFinal(encrypted);
        String originalString = new String(original);

        // Print the original decrypted string
        System.out.println("Decrypted text: " + originalString);

        // Close the scanner
        scanner.close();
    }
}
```

Output:



```
hii hello
```

Output:

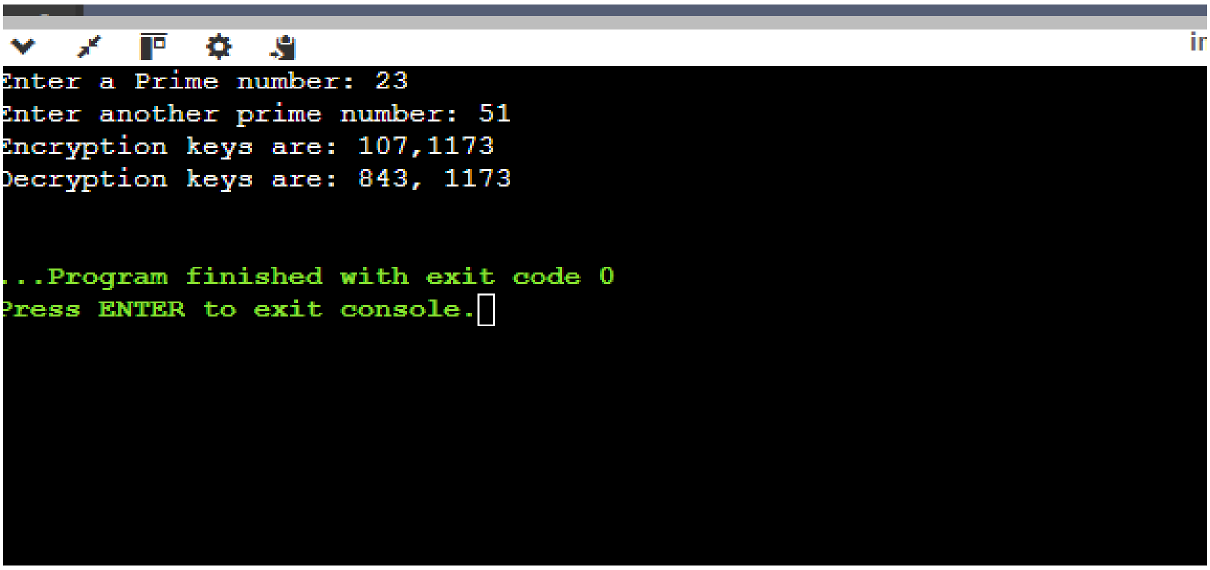
```
Input your message: Encrypted text (hex): 749a3300c19bd7c7010c2a9386910c71
Decrypted text: hii hello
```

Experiment-7

Write a Java program to implement RSA Algoithm.

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
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) {
        // TODO code application logic here
        System.out.print("Enter a Prime number: ");
        BigInteger p = sc.nextBigInteger(); // Here's one prime number..
        System.out.print("Enter another prime number: ");
        BigInteger q = sc.nextBigInteger(); // ..and another.
        BigInteger n = p.multiply(q);
        BigInteger n2 = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
        BigInteger e = generateE(n2);
        BigInteger d = e.modInverse(n2); // Here's the multiplicative inverse

        System.out.println("Encryption keys are: " + e + "," + n);
        System.out.println("Decryption keys are: " + d + ", " + n);
    }

    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;
    }
}
```

Output:



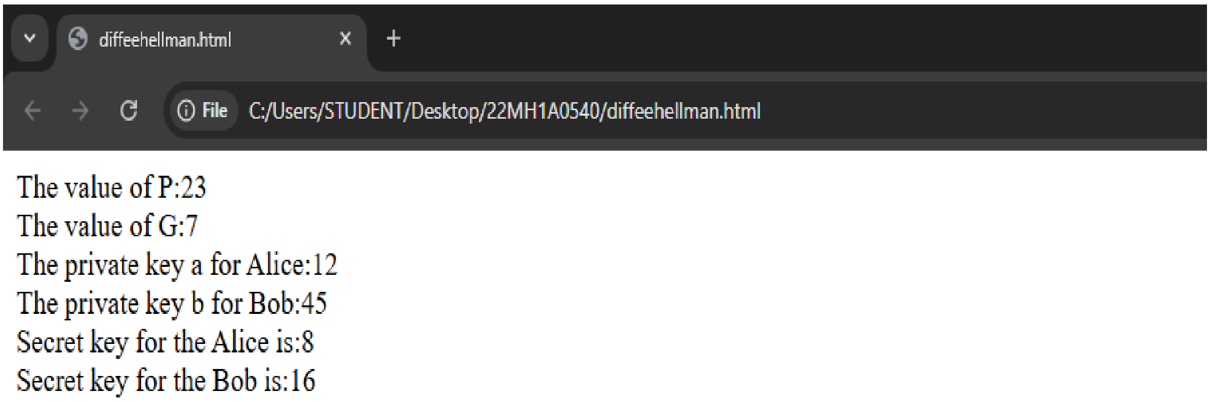
```
Enter a Prime number: 23
Enter another prime number: 51
Encryption keys are: 107,1173
Decryption keys are: 843, 1173


...Program finished with exit code 0
Press ENTER to exit console.
```

Experiment-8

Implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript. Consider the end user as one of the parties (Alice) and the JavaScript application as other party (bob).

```
<!DOCTYPE html>
<html>
<body>
<script>
function power(a, b, p)
{
 if (b == 1)
 return a;
 else
 return((Math.pow(a, b)) % p);
}
var P, G, x, a, y, b, ka, kb;
 var P = window.prompt("Enter value of P: ");
document.write("The value of P:" + P + "<br>");
 var G = window.prompt("Enter value of G: ");
document.write("The value of G:" + G + "<br>");
var a = window.prompt("Enter private key for Alice: ");
document.write("The private key a for Alice:" + a + "<br>");
x = power(G, a, P);
var b = window.prompt("Enter private key for bob: ");
document.write("The private key b for Bob:" +
 b + "<br>");
y = power(G, b, P);
ka = power(y, a, P);
kb = power(x, b, P);
document.write("Secret key for the Alice is:" +
 ka + "<br>");
document.write("Secret key for the Bob is:" +
 kb + "<br>");
</script>
</body>
</html>
```

Output:



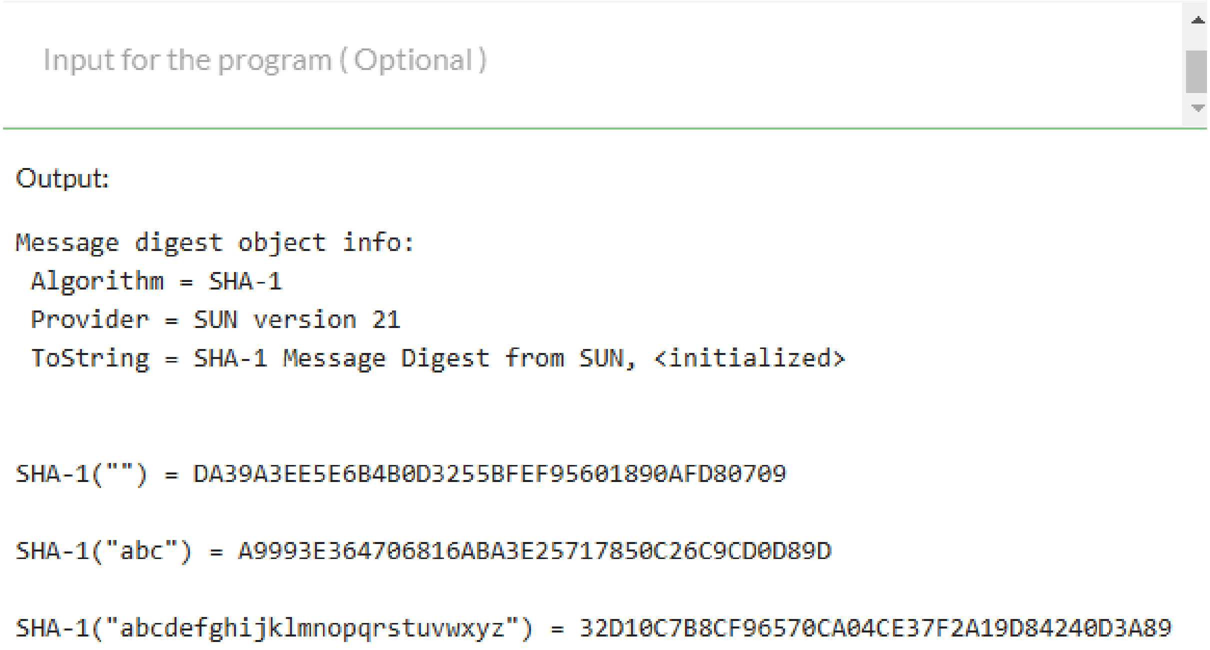
```
The value of P:23
The value of G:7
The private key a for Alice:12
The private key b for Bob:45
Secret key for the Alice is:8
Secret key for the Bob is:16
```

Experiment-9

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

```
import java.security.*;


public class SHA1 {
    public static void main(String[] args) {
        try {
            // Get a MessageDigest instance for SHA-1
            MessageDigest md = MessageDigest.getInstance("SHA-1");
            // Display MessageDigest object information
            System.out.println("Message digest object info: ");
            System.out.println(" Algorithm = " + md.getAlgorithm());
            System.out.println(" Provider = " + md.getProvider());
            System.out.println(" ToString = " + md.toString());
            // Example 1: Empty string
            String input = "";
            md.update(input.getBytes());
            byte[] output = md.digest();
            System.out.println();
            System.out.println("SHA-1(\"" + input + "\") = " + bytesToHex(output));
            // Example 2: String "abc"
            input = "abc";
            md.update(input.getBytes());
            output = md.digest();
            System.out.println();
            System.out.println("SHA-1(\"" + input + "\") = " + bytesToHex(output));
            // Example 3: String "abcdefghijklmnopqrstuvwxyz"
            input = "abcdefghijklmnopqrstuvwxyz";
            md.update(input.getBytes());
            output = md.digest();
            System.out.println();
            System.out.println("SHA-1(\"" + input + "\") = " + bytesToHex(output));
            System.out.println("");
        } catch (Exception e) {
            System.out.println("Exception: " + e);
        }
    }
    // Method to convert byte array to hexadecimal string
    public static String bytesToHex(byte[] b) {
        char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};
        StringBuffer buf = new StringBuffer();
        for (int j = 0; j < b.length; j++) {
            buf.append(hexDigit[(b[j] >> 4) & 0x0f]);
            buf.append(hexDigit[b[j] & 0x0f]);
        }
        return buf.toString();
    }
}
```

Output:



```
Message digest object info:
 Algorithm = SHA-1
 Provider = SUN version 21
 ToString = SHA-1 Message Digest from SUN, <initialized>


SHA-1("") = DA39A3EE5E6B4B0D3255BFEF95601890AFD80709

SHA-1("abc") = A9993E364706816ABA3E25717850C26C9CD0D89D

SHA-1("abcdefghijklmnopqrstuvwxyz") = 32D10C7B8CF96570CA04CE37F2A19D84240D3A89
```

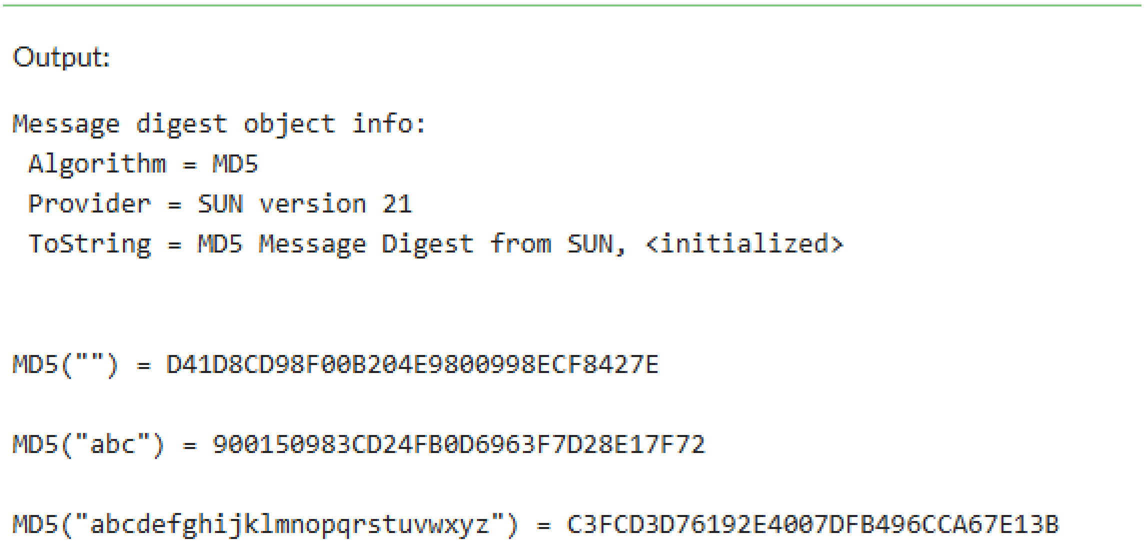
Experiment-10

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

```
import java.security.*;


public class MD5 {
   public static void main(String[] a) {
      // TODO code application logic here
      try {
         MessageDigest md = MessageDigest.getInstance("MD5");


         System.out.println("Message digest object info: ");
         System.out.println(" Algorithm = " + md.getAlgorithm());
         System.out.println(" Provider = " + md.getProvider());
         System.out.println(" ToString = " + md.toString());


         String input = "";
         md.update(input.getBytes());
         byte[] output = md.digest();
         System.out.println();
         System.out.println("MD5(\"" + input + "\") = " + bytesToHex(output));


         input = "abc";
         md.update(input.getBytes());
         output = md.digest();
         System.out.println();
         System.out.println("MD5(\"" + input + "\") = " + bytesToHex(output));


         input = "abcdefghijklmnopqrstuvwxyz";
         md.update(input.getBytes());
         output = md.digest();
         System.out.println();
         System.out.println("MD5(\"" + input + "\") = " + bytesToHex(output));
```

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


        } catch (Exception e) {
            System.out.println("Exception: " + e);
        }
    }
    public static String bytesToHex(byte[] b) {
        char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};
        StringBuffer buf = new StringBuffer();
        for (int j = 0; j < b.length; j++) {
            buf.append(hexDigit[(b[j] >> 4) & 0x0f]);
            buf.append(hexDigit[b[j] & 0x0f]);
        }
        return buf.toString();
    }
}
```

Output:



Output:

```
Message digest object info:
 Algorithm = MD5
 Provider = SUN version 21
 ToString = MD5 Message Digest from SUN, <initialized>


MD5("") = D41D8CD98F00B204E9800998ECF8427E

MD5("abc") = 900150983CD24FB0D6963F7D28E17F72

MD5("abcdefghijklmnopqrstuvwxyz") = C3FCD3D76192E4007DFB496CCA67E13B
```