Date: ……………………….

EXPERIMENT 1:

Aim: Write a JAVA program to implement the DES algorithm logic.

Program:

```
                    import java.util.*;

        class DES {
                    private static final byte[] IP = {
                                58, 50, 42, 34, 26, 18, 10, 2,
                                60, 52, 44, 36, 28, 20, 12, 4,
                                62, 54, 46, 38, 30, 22, 14, 6,
                                64, 56, 48, 40, 32, 24, 16, 8,
                                57, 49, 41, 33, 25, 17, 9,  1,
                                59, 51, 43, 35, 27, 19, 11, 3,
                                61, 53, 45, 37, 29, 21, 13, 5,
                                63, 55, 47, 39, 31, 23, 15, 7
                    };
                    private static final byte[] PC1 = {
                                57, 49, 41, 33, 25, 17, 9,
                                1,  58, 50, 42, 34, 26, 18,
                                10, 2,  59, 51, 43, 35, 27,
                                19, 11, 3,  60, 52, 44, 36,
                                63, 55, 47, 39, 31, 23, 15,
                                7,  62, 54, 46, 38, 30, 22,
                                14, 6,  61, 53, 45, 37, 29,
                                21, 13, 5,  28, 20, 12, 4
                    };
                    private static final byte[] PC2 = {
                                14, 17, 11, 24, 1,  5,
                                3,  28, 15, 6,  21, 10,
                                23, 19, 12, 4,  26, 8,
                                16, 7,  27, 20, 13, 2,
                                41, 52, 31, 37, 47, 55,
                                30, 40, 51, 45, 33, 48,
                                44, 49, 39, 56, 34, 53,
                                46, 42, 50, 36, 29, 32
                    };
                    private static final byte[] rotations = {
                                1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1
                    };
                    private static final byte[] E = {
                                32, 1,  2,  3,  4,  5,
                                4,  5,  6,  7,  8,  9,
                                8,  9,  10, 11, 12, 13,
```

Date: ……………………….

```
            12, 13, 14, 15, 16, 17,
            16, 17, 18, 19, 20, 21,
            20, 21, 22, 23, 24, 25,
            24, 25, 26, 27, 28, 29,
            28, 29, 30, 31, 32, 1
};
private static final byte[][] S = { {
            14, 4,  13, 1,  2,  15, 11, 8,  3,  10, 6,  12, 5,  9,  0,  7,
            0,  15, 7,  4,  14, 2,  13, 1,  10, 6,  12, 11, 9,  5,  3,  8,
            4,  1,  14, 8,  13, 6,  2,  11, 15, 12, 9,  7,  3,  10, 5,  0,
            15, 12, 8,  2,  4,  9,  1,  7,  5,  11, 3,  14, 10, 0,  6,  13
}, {
            15, 1,  8,  14, 6,  11, 3,  4,  9,  7,  2,  13, 12, 0,  5,  10,
            3,  13, 4,  7,  15, 2,  8,  14, 12, 0,  1,  10, 6,  9,  11, 5,
            0,  14, 7,  11, 10, 4,  13, 1,  5,  8,  12, 6,  9,  3,  2,  15,
            13, 8,  10, 1,  3,  15, 4,  2,  11, 6,  7,  12, 0,  5,  14, 9
}, {
            10, 0,  9,  14, 6,  3,  15, 5,  1,  13, 12, 7,  11, 4,  2,  8,
            13, 7,  0,  9,  3,  4,  6,  10, 2,  8,  5,  14, 12, 11, 15, 1,
            13, 6,  4,  9,  8,  15, 3,  0,  11, 1,  2,  12, 5,  10, 14, 7,
            1,  10, 13, 0,  6,  9,  8,  7,  4,  15, 14, 3,  11, 5,  2,  12
}, {
            7,  13, 14, 3,  0,  6,  9,  10, 1,  2,  8,  5,  11, 12, 4,  15,
            13, 8,  11, 5,  6,  15, 0,  3,  4,  7,  2,  12, 1,  10, 14, 9,
            10, 6,  9,  0,  12, 11, 7,  13, 15, 1,  3,  14, 5,  2,  8,  4,
            3,  15, 0,  6,  10, 1,  13, 8,  9,  4,  5,  11, 12, 7,  2,  14
}, {
            2,  12, 4,  1,  7,  10, 11, 6,  8,  5,  3,  15, 13, 0,  14, 9,
            14, 11, 2,  12, 4,  7,  13, 1,  5,  0,  15, 10, 3,  9,  8,  6,
            4,  2,  1,  11, 10, 13, 7,  8,  15, 9,  12, 5,  6,  3,  0,  14,
            11, 8,  12, 7,  1,  14, 2,  13, 6,  15, 0,  9,  10, 4,  5,  3
}, {
            12, 1,  10, 15, 9,  2,  6,  8,  0,  13, 3,  4,  14, 7,  5,  11,
            10, 15, 4,  2,  7,  12, 9,  5,  6,  1,  13, 14, 0,  11, 3,  8,
            9,  14, 15, 5,  2,  8,  12, 3,  7,  0,  4,  10, 1,  13, 11, 6,
            4,  3,  2,  12, 9,  5,  15, 10, 11, 14, 1,  7,  6,  0,  8,  13
}, {
            4,  11, 2,  14, 15, 0,  8,  13, 3,  12, 9,  7,  5,  10, 6,  1,
            13, 0,  11, 7,  4,  9,  1,  10, 14, 3,  5,  12, 2,  15, 8,  6,
            1,  4,  11, 13, 12, 3,  7,  14, 10, 15, 6,  8,  0,  5,  9,  2,
            6,  11, 13, 8,  1,  4,  10, 7,  9,  5,  0,  15, 14, 2,  3,  12
}, {
            13, 2,  8,  4,  6,  15, 11, 1,  10, 9,  3,  14, 5,  0,  12, 7,
            1,  15, 13, 8,  10, 3,  7,  4,  12, 5,  6,  11, 0,  14, 9,  2,
            7,  11, 4,  1,  9,  12, 14, 2,  0,  6,  10, 13, 15, 3,  5,  8,
            2,  1,  14, 7,  4,  10, 8,  13, 15, 12, 9,  0,  3,  5,  6,  11
} };
```

```
private static final byte[] P = {
        16, 7,  20, 21,
        29, 12, 28, 17,
        1,  15, 23, 26,
        5,  18, 31, 10,
        2,  8,  24, 14,
        32, 27, 3,  9,
        19, 13, 30, 6,
        22, 11, 4,  25
};
private static final byte[] FP = {
        40, 8, 48, 16, 56, 24, 64, 32,
        39, 7, 47, 15, 55, 23, 63, 31,
        38, 6, 46, 14, 54, 22, 62, 30,
        37, 5, 45, 13, 53, 21, 61, 29,
        36, 4, 44, 12, 52, 20, 60, 28,
        35, 3, 43, 11, 51, 19, 59, 27,
        34, 2, 42, 10, 50, 18, 58, 26,
        33, 1, 41, 9, 49, 17, 57, 25
};
private static int[] C = new int[28];
private static int[] D = new int[28];
private static int[][] subkey = new int[16][48];
public static void main(String args[]) {
        System.out.println("Enter the input as a 16 character hexadecimal value:");
        String input = new Scanner(System.in).nextLine();
        int inputBits[] = new int[64];

        for(int i=0 ; i < 16 ; i++) {

                String s = Integer.toBinaryString(Integer.parseInt(input.charAt(i) + "", 16));


                while(s.length() < 4) {
                        s = "0" + s;
                }

                for(int j=0 ; j < 4 ; j++) {
                        inputBits[(4*i)+j] = Integer.parseInt(s.charAt(j) + "");
                }
        }

        System.out.println("Enter the key as a 16 character hexadecimal value:");
        String key = new Scanner(System.in).nextLine();
        int keyBits[] = new int[64];
        for(int i=0 ; i < 16 ; i++) {
                String s = Integer.toBinaryString(Integer.parseInt(key.charAt(i) + "", 16));
```

Date: ……………………….

```
				while(s.length() < 4) {
					s = "0" + s;
				}
				for(int j=0 ; j < 4 ; j++) {
					keyBits[(4*i)+j] = Integer.parseInt(s.charAt(j) + "");
				}
			}
			System.out.println("\n+++ ENCRYPTION +++");
			int outputBits[] = permute(inputBits, keyBits, false);
			System.out.println("\n+++ DECRYPTION +++");
			permute(outputBits, keyBits, true);
		}
		private static int[] permute(int[] inputBits, int[] keyBits, boolean isDecrypt) {

			int newBits[] = new int[inputBits.length];
			for(int i=0 ; i < inputBits.length ; i++) {
				newBits[i] = inputBits[IP[i]-1];
			}
			int L[] = new int[32];
			int R[] = new int[32];
			int i;

			for(i=0 ; i < 28 ; i++) {
				C[i] = keyBits[PC1[i]-1];
			}
			for( ; i < 56 ; i++) {
				D[i-28] = keyBits[PC1[i]-1];
			}

			System.arraycopy(newBits, 0, L, 0, 32);
			System.arraycopy(newBits, 32, R, 0, 32);
			System.out.print("\nL0 = ");
			displayBits(L);
			System.out.print("R0 = ");
			displayBits(R);
			for(int n=0 ; n < 16 ; n++) {
				System.out.println("\n-------------");
				System.out.println("Round " + (n+1) + ":");

				int newR[] = new int[0];
				if(isDecrypt) {
					newR = fiestel(R, subkey[15-n]);
					System.out.print("Round key = ");
					displayBits(subkey[15-n]);
				} else {
					newR = fiestel(R, KS(n, keyBits));
					System.out.print("Round key = ");
					displayBits(subkey[n]);
```

```
                }

                int newL[] = xor(L, newR);
                L = R;
                R = newL;
                System.out.print("L = ");
                displayBits(L);
                System.out.print("R = ");
                displayBits(R);
        }
        int output[] = new int[64];
        System.arraycopy(R, 0, output, 0, 32);
        System.arraycopy(L, 0, output, 32, 32);
        int finalOutput[] = new int[64];

        for(i=0 ; i < 64 ; i++) {
                finalOutput[i] = output[FP[i]-1];
        }

        String hex = new String();
        for(i=0 ; i < 16 ; i++) {
                String bin = new String();
                for(int j=0 ; j < 4 ; j++) {
                        bin += finalOutput[(4*i)+j];
                }
                int decimal = Integer.parseInt(bin, 2);
                hex += Integer.toHexString(decimal);
        }
        if(isDecrypt) {
                System.out.print("Decrypted text: ");

        } else {
                System.out.print("Encrypted text: ");
        }
        System.out.println(hex.toUpperCase());
        return finalOutput;
}
private static int[] KS(int round, int[] key) {

        int C1[] = new int[28];
        int D1[] = new int[28];

int rotationTimes = (int) rotations[round];

        C1 = leftShift(C, rotationTimes);
        D1 = leftShift(D, rotationTimes);

        int CnDn[] = new int[56];
```

```
		System.arraycopy(C1, 0, CnDn, 0, 28);
		System.arraycopy(D1, 0, CnDn, 28, 28);

		int Kn[] = new int[48];
		for(int i=0 ; i < Kn.length ; i++) {
			Kn[i] = CnDn[PC2[i]-1];
		}

		subkey[round] = Kn;
		C = C1;
		D = D1;
		return Kn;
	}
	private static int[] fiestel(int[] R, int[] roundKey) {

		int expandedR[] = new int[48];
		for(int i=0 ; i < 48 ; i++) {
			expandedR[i] = R[E[i]-1];
		}

		int temp[] = xor(expandedR, roundKey);

		int output[] = sBlock(temp);
		return output;
	}
	private static int[] xor(int[] a, int[] b) {

		int answer[] = new int[a.length];
		for(int i=0 ; i < a.length ; i++) {
			answer[i] = a[i]^b[i];
		}
		return answer;
	}
	private static int[] sBlock(int[] bits) {

		int output[] = new int[32];

		for(int i=0 ; i < 8 ; i++) {

			int row[] = new int [2];
			row[0] = bits[6*i];
			row[1] = bits[(6*i)+5];
			String sRow = row[0] + "" + row[1];

			int column[] = new int[4];
			column[0] = bits[(6*i)+1];
			column[1] = bits[(6*i)+2];
			column[2] = bits[(6*i)+3];
```

```
                        column[3] = bits[(6*i)+4];
                        String sColumn = column[0] +""+ column[1] +""+ column[2] +""+ column[3];

                        int iRow = Integer.parseInt(sRow, 2);
                        int iColumn = Integer.parseInt(sColumn, 2);
                        int x = S[i][(iRow*16) + iColumn];

                        String s = Integer.toBinaryString(x);

                        while(s.length() < 4) {
                                s = "0" + s;
                        }

                        for(int j=0 ; j < 4 ; j++) {
                                output[(i*4) + j] = Integer.parseInt(s.charAt(j) + "");
                        }
                }
                int finalOutput[] = new int[32];
                for(int i=0 ; i < 32 ; i++) {
                        finalOutput[i] = output[P[i]-1];
                }
                return finalOutput;
        }
        private static int[] leftShift(int[] bits, int n) {

                int answer[] = new int[bits.length];
                System.arraycopy(bits, 0, answer, 0, bits.length);
                for(int i=0 ; i < n ; i++) {
                        int temp = answer[0];
                        for(int j=0 ; j < bits.length-1 ; j++) {
                                answer[j] = answer[j+1];
                        }
                        answer[bits.length-1] = temp;
                }
                return answer;
        }
        private static void displayBits(int[] bits) {

                for(int i=0 ; i < bits.length ; i+=4) {
                        String output = new String();
                        for(int j=0 ; j < 4 ; j++) {
                                output += bits[i+j];
                        }
                        System.out.print(Integer.toHexString(Integer.parseInt(output, 2)));
                }
                System.out.println();
        }
}
```

Date: ……………………….

Output:

```
Enter the input as a 16 character hexadecimal value:
123456ABCD132536
Enter the key as a 16 character hexadecimal value:
AABB09182736CCDD

+++ ENCRYPTION +++

L0 = 14a7d678
R0 = 18ca18ad

-------------
Round 1:
Round key = 194cd072de8c
L = 18ca18ad
R = 5a78e394

-------------
Round 2:
Round key = 4568581abcce
L = 5a78e394
R = 4a1210f6

-------------
Round 3:
Round key = 06eda4acf5b5
L = 4a1210f6
R = b8089591

-------------
Round 4:
Round key = da2d032b6ee3
L = b8089591
R = 236779c2

-------------
Round 5:
Round key = 69a629fec913
L = 236779c2
R = a15a4b87

-------------
Round 6:
```

Date: ……………………….

```
Round key = c1948e87475e
L = a15a4b87
R = 2e8f9c65

-------------
Round 7:
Round key = 708ad2ddb3c0
L = 2e8f9c65
R = a9fc20a3

-------------
Round 8:
Round key = 34f822f0c66d
L = a9fc20a3
R = 308bee97

-------------
Round 9:
Round key = 84bb4473dccc
L = 308bee97
R = 10af9d37

-------------
Round 10:
Round key = 02765708b5bf
L = 10af9d37
R = 6ca6cb20

-------------
Round 11:
Round key = 6d5560af7ca5
L = 6ca6cb20
R = ff3c485f

-------------
Round 12:
Round key = c2c1e96a4bf3
L = ff3c485f
R = 22a5963b

-------------
Round 13:
Round key = 99c31397c91f
L = 22a5963b
```

Date: ……………………….

```
R = 387ccdaa

-------------
Round 14:
Round key = 251b8bc717d0
L = 387ccdaa
R = bd2dd2ab

-------------
Round 15:
Round key = 3330c5d9a36d
L = bd2dd2ab
R = cf26b472

-------------
Round 16:
Round key = 181c5d75c66d
L = cf26b472
R = 19ba9212
Encrypted text: C0B7A8D05F3A829C

+++ DECRYPTION +++

L0 = 19ba9212
R0 = cf26b472

-------------
Round 1:
Round key = 181c5d75c66d
L = cf26b472
R = bd2dd2ab

-------------
Round 2:
Round key = 3330c5d9a36d
L = bd2dd2ab
R = 387ccdaa

-------------
Round 3:
Round key = 251b8bc717d0
L = 387ccdaa
R = 22a5963b
```

Date: ……………………….

```
-------------
Round 4:
Round key = 99c31397c91f
L = 22a5963b
R = ff3c485f

-------------
Round 5:
Round key = c2c1e96a4bf3
L = ff3c485f
R = 6ca6cb20

-------------
Round 6:
Round key = 6d5560af7ca5
L = 6ca6cb20
R = 10af9d37

-------------
Round 7:
Round key = 02765708b5bf
L = 10af9d37
R = 308bee97

-------------
Round 8:
Round key = 84bb4473dccc
L = 308bee97
R = a9fc20a3

-------------
Round 9:
Round key = 34f822f0c66d
L = a9fc20a3
R = 2e8f9c65

-------------
Round 10:
Round key = 708ad2ddb3c0
L = 2e8f9c65
R = a15a4b87

-------------
Round 11:
```

Date: ……………………….

```
Round key = c1948e87475e
L = a15a4b87
R = 236779c2

-------------
Round 12:
Round key = 69a629fec913
L = 236779c2
R = b8089591

-------------
Round 13:
Round key = da2d032b6ee3
L = b8089591
R = 4a1210f6

-------------
Round 14:
Round key = 06eda4acf5b5
L = 4a1210f6
R = 5a78e394

-------------
Round 15:
Round key = 4568581abcce
L = 5a78e394
R = 18ca18ad

-------------
Round 16:
Round key = 194cd072de8c
L = 18ca18ad
R = 14a7d678
Decrypted text: 123456ABCD132536
```

Date: ……………………….

EXPERIMENT 2:

Aim: Write a Java program that contains functions, which accept a key and input text to be encrypted/decrypted. This program should use the key to encrypt/decrypt the input by using the triple Des algorithm. Make use of Java Cryptography package.

Program:

```
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 sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;
public class TDES {
private static final String UNICODE_FORMAT = "UTF8";
public static final String DESEDE_ENCRYPTION_SCHEME = "DESede";
private KeySpecmyKeySpec;
private SecretKeyFactorymySecretKeyFactory;
private Cipher cipher;
byte[] keyAsBytes;
private String myEncryptionKey;
private String myEncryptionScheme;
SecretKey key;
static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
public TDES() 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) {
String encryptedString = null;
try {
cipher.init(Cipher.ENCRYPT_MODE, key);
byte[] plainText = unencryptedString.getBytes(UNICODE_FORMAT);
byte[] encryptedText = cipher.doFinal(plainText);
BASE64Encoder base64encoder = new BASE64Encoder();
encryptedString = base64encoder.encode(encryptedText); }
catch (Exception e) {
e.printStackTrace(); }
return encryptedString; }
public String decrypt(String encryptedString) {
String decryptedText=null;
try {
cipher.init(Cipher.DECRYPT_MODE, key);
BASE64Decoder base64decoder = new BASE64Decoder();
byte[] encryptedText = base64decoder.decodeBuffer(encryptedString);
byte[] plainText = cipher.doFinal(encryptedText);
decryptedText= bytes2String(plainText); }
catch (Exception e) {
e.printStackTrace(); }
return decryptedText; }
private static String bytes2String(byte[] bytes) {
StringBufferstringBuffer = new StringBuffer();
for (int i = 0; i <bytes.length; i++) {
stringBuffer.append((char) bytes[i]); }
returnstringBuffer.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(""); } }
```

Date: ……………………….

Output:

```
 Enter the string:    Welcome
String To Encrypt:   Welcome
Encrypted Value :    BPQMwc0wKvg=
Decrypted Value :    Welcome
```

Date: ……………………….

EXPERIMENT 3:

Aim: Write a JAVA program to implement the Blowfish algorithm logic.

Program:

```
  import java.io.*;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.security.Key;
import javax.crypto.Cipher;
import javax.crypto.CipherOutputStream;
import javax.crypto.KeyGenerator;
import sun.misc.BASE64Encoder;
public class BlowFish {
public static void main(String[] args) throws Exception {
KeyGenerator keyGenerator = KeyGenerator.getInstance("Blowfish");
keyGenerator.init(128);
Key secretKey = keyGenerator.generateKey();
Cipher cipherOut = Cipher.getInstance("Blowfish/CFB/NoPadding");
cipherOut.init(Cipher.ENCRYPT_MODE, secretKey);
BASE64Encoder encoder = new BASE64Encoder();
byte iv[] = cipherOut.getIV();
if (iv != null) {
System.out.println("Initialization Vector of the Cipher: " + encoder.encode(iv)); }
FileInputStream fin = new FileInputStream("inputFile.txt");
FileOutputStream fout = new FileOutputStream("outputFile.txt");
CipherOutputStream cout = new CipherOutputStream(fout, cipherOut);
int input = 0;
while ((input = fin.read()) != -1) {
cout.write(input); }
fin.close(); cout.close(); } }
```

Date: ……………………….

Output:

```
Initialization Vector of the Cipher: dI1MXzW97oQ=
Contents of inputFile.txt: Hello World
Contents of outputFile.txt: ùJÖ˜ NåI“
```

Date: ……………………….

EXPERIMENT 4:

Aim: Using Java cryptography, encrypt the text “Hello world” using Blowfish. Create your own key using Java key tool.

Program:

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.swing.JOptionPane;
public class BlowFishCipher {
public static void main(String[] args) throws Exception {
// create a key generator based upon the Blowfish cipher
KeyGenerator keygenerator = KeyGenerator.getInstance("Blowfish");
// create a key
SecretKey secretkey = keygenerator.generateKey();
// create a cipher based upon Blowfish
Cipher cipher = Cipher.getInstance("Blowfish");
// initialise cipher to with secret key
cipher.init(Cipher.ENCRYPT_MODE, secretkey);
// get the text to encrypt
String inputText = JOptionPane.showInputDialog("Input your message: ");
// encrypt message
byte[] encrypted = cipher.doFinal(inputText.getBytes());
// re-initialise the cipher to be in decrypt mode
cipher.init(Cipher.DECRYPT_MODE, secretkey);
// decrypt message
byte[] decrypted = cipher.doFinal(encrypted);
// and display the results
JOptionPane.showMessageDialog(JOptionPane.getRootFrame(),
"\nEncrypted text: " + new String(encrypted) + "\n" +
"\nDecrypted text: " + new String(decrypted));
System.exit(0);
} }
```

Output:

```
Input your message: Hello world
Encrypted text: 3ooo&&(*&*4r4
Decrypted text: Hello world
```

Date: ……………………….

EXPERIMENT 5:

Aim: Write a Java program to implement RSA algorithm.

Program:

```
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) {
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: 3
Enter another prime number: 5
Encryption keys are: 5, 15
Decryption keys are: 5, 15
```

Date: ……………………….

EXPERIMENT 6:

Aim: 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 the other party(Bob).

Program:

```
import java.math.BigInteger;
import java.security.KeyFactory;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.SecureRandom;
import javax.crypto.spec.DHParameterSpec;
import javax.crypto.spec.DHPublicKeySpec;
public class DiffeHellman {
public final static int pValue = 47;
public final static int gValue = 71;
public final static int XaValue = 9;
public final static int XbValue = 14;
public static void main(String[] args) throws Exception {
BigInteger p = new BigInteger(Integer.toString(pValue));
BigInteger g = new BigInteger(Integer.toString(gValue));
BigInteger Xa = new BigInteger(Integer.toString(XaValue));
```

Date: ……………………….

```
BigInteger Xb = new BigInteger(Integer.toString(XbValue));
createKey();
int bitLength = 512; // 512 bits
SecureRandom rnd = new SecureRandom();
p = BigInteger.probablePrime(bitLength, rnd);
g = BigInteger.probablePrime(bitLength, rnd);
createSpecificKey(p, g);
}
public static void createKey() throws Exception {
KeyPairGenerator kpg = KeyPairGenerator.getInstance("DiffieHellman");
kpg.initialize(512);
KeyPair kp = kpg.generateKeyPair();
KeyFactory kfactory = KeyFactory.getInstance("DiffieHellman");
DHPublicKeySpec kspec =
(DHPublicKeySpec)kfactory.getKeySpec(kp.getPublic(),DHPublicKeySpec.class
);
System.out.println("Public key is: " + kspec);
}
public static void createSpecificKey(BigInteger p, BigInteger g) throws
Exception {
KeyPairGenerator kpg = KeyPairGenerator.getInstance("DiffieHellman");
DHParameterSpec param = new DHParameterSpec(p, g);
kpg.initialize(param);
KeyPair kp = kpg.generateKeyPair();
KeyFactory kfactory = KeyFactory.getInstance("DiffieHellman");
```

Date: ……………………….

```
DHPublicKeySpec kspec =
(DHPublicKeySpec)kfactory.getKeySpec(kp.getPublic(),DHPublicKeySpec.class
);

System.out.println("\nPublic key is : " + kspec);

}

}
```

Output:

```
Public key is: javax.crypto.spec.DHPublicKeySpec@372f7a8d

Public key is : javax.crypto.spec.DHPublicKeySpec@2f92e0f4
```

Date: ……………………….

EXPERIMENT 7:

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

Program:

```
import java.security.*;
public class SHA1 {
public static void main(String[] a) {
try {
MessageDigest md = MessageDigest.getInstance("SHA1");
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("SHA1(\""+input+"\") = " +bytesToHex(output));
input = "abc";
md.update(input.getBytes());
output = md.digest();
System.out.println();
System.out.println("SHA1(\""+input+"\") = " +bytesToHex(output));
input = "abcdefghijklmnopqrstuvwxyz";
```

Date: ……………………….

```
md.update(input.getBytes());
output = md.digest();
System.out.println();
System.out.println("SHA1(\"" +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:

```
              Message digest object info:
    Algorithm = SHA1
    Provider = SUN version 1.8
    ToString = SHA1 Message Digest from SUN, <initialized>

SHA1("") = DA39A3EE5E6B4B0D3255BFEF95601890AFD80709
SHA1("abc") = A9993E364706816ABA3E25717850C26C9CD0D89D
SHA1("abcdefghijklmnopqrstuvwxyz") =  32D10C7B8CF96570CA04CE37F2A19D84240D3A89
```

Date: ……………………….

EXPERIMENT 8:

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

Program:

```
import java.security.*;
public class MD5 {
public static void main(String[] a) {
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:

```
      Message digest object info:
     Algorithm = MD5
     Provider = SUN version 1.8
     ToString = MD5 Message Digest from SUN, <initialized>

MD5("") = D41D8CD98F00B204E9800998ECF8427E

MD5("abc") = 900150983CD24FB0D6963F7D28E17F72

MD5("abcdefghijklmnopqrstuvwxyz") = C3FCD3D76192E4007DFB496CCA67E13B
```

EXPERIMENT 9:

Aim: Explore the Java classes related to digital certificates.

For the purposes of digital signing of documents, verification of digital signatures, and handling digital certificates in the Java platform, the Java Cryptography Architecture (JCA) is used. JCA is a specification that gives the programmers a standard way to access cryptographic services, digital signatures, and digital certificates.

The JCA provides classes and interfaces for working with public and private keys, digital certificates, message signing, digital signatures verification, accessing protected keystores, and some other processes. These classes and interfaces are located in the standard packages java.security and java.security.cert

The Most Important Classes in JCA

java.security.KeyStore—gives access to protected keystores for certificates and passwords. The keystores are represented as set of entries and each entry has a unique name, called an alias. The KeyStore class has methods for loading keystore from a stream, storing a keystore to a stream, enumerating the entries in the keystore, extracting keys, certificates and certification chains, modifying entries in the keystore, and so forth. Two major formats for storing keystores are supported—PFX (according to the PKCS#12 standard) and JKS (Java Key Store format used by JDK internally). When we create objects of the class KeyStore, the format of the keystore should be given as a parameter. The possible values are "JKS" and "PKCS12". Objects stored in a keystore can be accessed by the alias but for accessing keys a password also is required.

java.security.PublicKey—represents a public key. It holds the key itself, its encoding format, and the algorithm destined to be used with this key.

java.security.PrivateKey—represents a private key. It holds the key itself, its encoding format, and the algorithm destined to be used with this key.

java.security.cert.Certificate—it is an abstract class for all classes that represent digital certificates. It contains a public key and information for its owner. For representing each particular type of certificates (for example X.509, PGP, and so forth), an appropriate derived class is used.

java.security.cert.X509Certificate—represents an X.509 v.3 certificate. It provides methods for accessing its attributes—owner (Subject), issuer, public key of the owner, period of validity, version, serial number, digital signature algorithm, digital signature, additional extensions, and so forth. All the information in an X509Certificate object is available for reading only.

java.security.cert.CertificateFactory—provides functionality for loading certificates, certification chains, and CRL lists from a stream. The generateCertificate() method that is purposed for reading a certificate from a stream expects the certificate to be DER-encoded (according to the PKCS#7 standard) and to be in a binary or text format (Base64-encoded). For reading a certification chain, the generateCertPath() method can be used and the encoding for the chain can be specified. Acceptable are encodings such as PkiPath, that corresponds to the ASN.1 DER sequence of certificates, and PKCS7 that corresponds to the PKCS#7 SignedData object (usually, such objects are stored in files with the standard extension .P7B). It is important to take into account the fact that PKCS7 encoding does not preserve the order of the certificates and, due to this particularity, we cannot use it for storing and reading certification chains. In Java, the only standard encoding for staring certification chains is PkiPath.

java.security.GeneralSecurityException & java.security.cert.CertificateException are classes for exceptions that can be thrown when working with digital signatures and certificates.

Date: ……………………….

EXPERIMENT 10:

Aim: Create a digital certificate of your own by using the Java key tool.

Requirements:

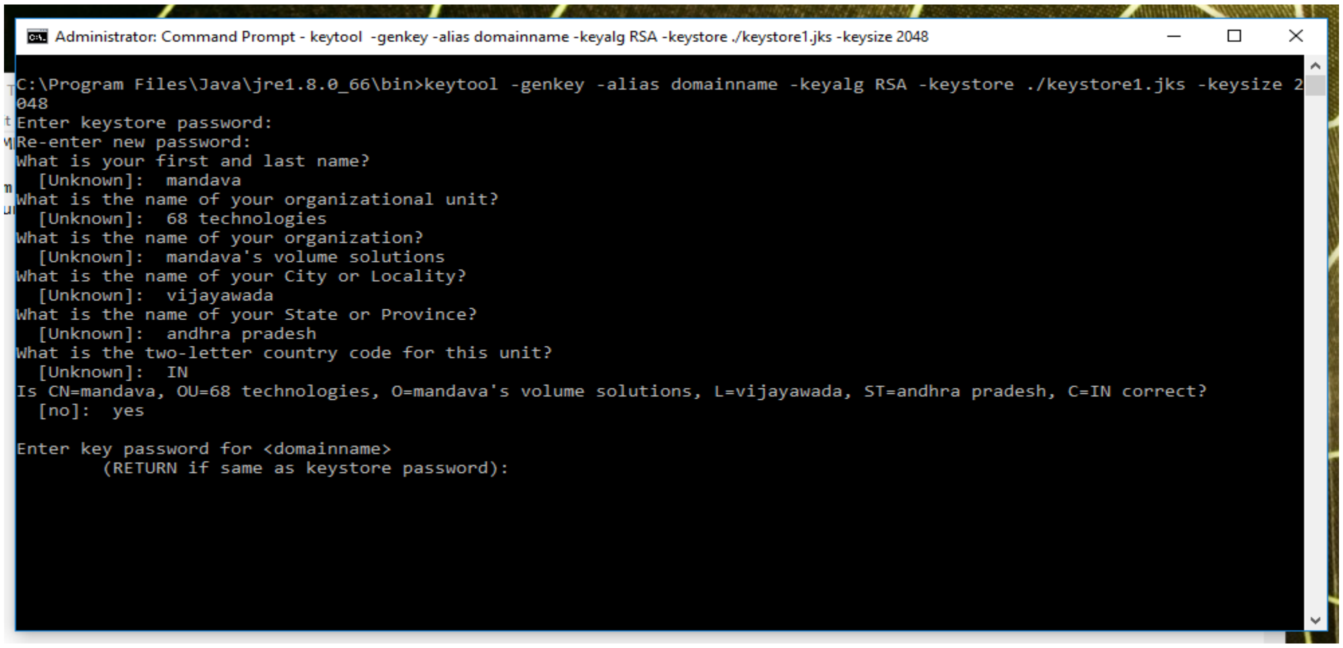
- JAVA software should be installed in your pc/laptop environment

Steps to create digital certificate:

- For windows users, the key tool path is
  C:\Program Files\Java\jre1.8.0_66\bin>
  For ubuntu/Linux users, key tool path is
  JAVA_HOME\bin\
- Now open cmd in the following path
  C:\Program Files\Java\jre1.8.0_66\bin>
  For ubuntu/Linux users, open terminal in the following path
  JAVA_HOME\bin\
- Now enter the following command

```
keytool -genkey -alias domainname -keyalg RSA -keystore ./keystore1.jks -keysize 2048
```

- After you entering the above command you must give some information related to your certificate like below screen shot.


Administrator: Command Prompt - keytool -genkey -alias domainname -keyalg RSA -keystore ./keystore1.jks -keysize 2048

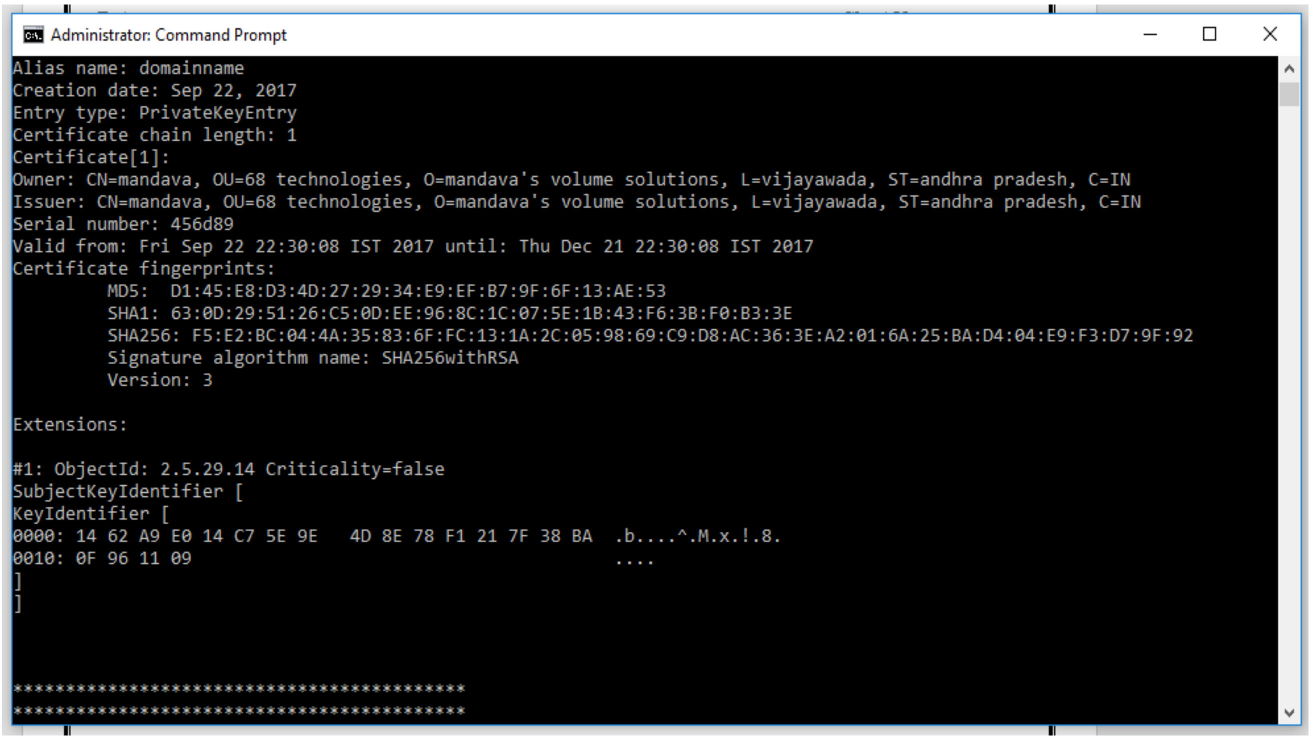
```
C:\Program Files\Java\jre1.8.0_66\bin>keytool -genkey -alias domainname -keyalg RSA -keystore ./keystore1.jks -keysize 2
048
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  mandava
What is the name of your organizational unit?
  [Unknown]:  68 technologies
What is the name of your organization?
  [Unknown]:  mandava's volume solutions
What is the name of your City or Locality?
  [Unknown]:  vijayawada
What is the name of your State or Province?
  [Unknown]:  andhra pradesh
What is the two-letter country code for this unit?
  [Unknown]:  IN
Is CN=mandava, OU=68 technologies, O=mandava's volume solutions, L=vijayawada, ST=andhra pradesh, C=IN correct?
  [no]:  yes

Enter key password for <domainname>
        (RETURN if same as keystore password):
```

- After entering all the details, you must type "YES "and enter then you must again give the password which is given above for key store.
- Now to view your created certificate you must enter the following command

C:\Program Files\Java\jre1.8.0_66\bin >keytool -list -v -keystore keystore1.jks

Output:

Now you will get the certificate as below in the screenshot with the information provided by you.



```
Alias name: domainname
Creation date: Sep 22, 2017
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=mandava, OU=68 technologies, O=mandava's volume solutions, L=vijayawada, ST=andhra pradesh, C=IN
Issuer: CN=mandava, OU=68 technologies, O=mandava's volume solutions, L=vijayawada, ST=andhra pradesh, C=IN
Serial number: 456d89
Valid from: Fri Sep 22 22:30:08 IST 2017 until: Thu Dec 21 22:30:08 IST 2017
Certificate fingerprints:
         MD5:  D1:45:E8:D3:4D:27:29:34:E9:EF:B7:9F:6F:13:AE:53
         SHA1: 63:0D:29:51:26:C5:0D:EE:96:8C:1C:07:5E:1B:43:F6:3B:F0:B3:3E
         SHA256: F5:E2:BC:04:4A:35:83:6F:FC:13:1A:2C:05:98:69:C9:D8:AC:36:3E:A2:01:6A:25:BA:D4:04:E9:F3:D7:9F:92
         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 14 62 A9 E0 14 C7 5E 9E   4D 8E 78 F1 21 7F 38 BA  .b....^.M.x.!.8.
0010: 0F 96 11 09                                        ....
]
]



*******************************************
*******************************************
```

Date: ……………………….

EXPERIMENT 12:

Aim: Write a program in java, which performs a digital signature on a given text.

Program:

```
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.Signature;
import sun.misc.BASE64Encoder;
public class DigSign {
public static void main(String[] args) throws Exception {
// TODO code application logic here
KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");
kpg.initialize(1024);
KeyPair keyPair = kpg.genKeyPair();
byte[] data = "Sample Text".getBytes("UTF8");
Signature sig = Signature.getInstance("MD5WithRSA");
sig.initSign(keyPair.getPrivate());
sig.update(data);
byte[] signatureBytes = sig.sign();
System.out.println("Signature: \n" + new
BASE64Encoder().encode(signatureBytes));
sig.initVerify(keyPair.getPublic());
sig.update(data);
System.out.println(sig.verify(signatureBytes));
}
}
```

Date: ……………………….

Output:

```
Signature:
b9LojjoRIPPdnAsz0Q/8WU4DyAsMYCWmE/c9oIZPutzw2SOhDKMOY7O/Ifb1sCRjXxPGo
UmMyrjE
Sss6yRXkMh6jTs0qBALSPWxzfGQ2ZT5WiwcDi+2ipWdjQRAAGJEDixA4fZ2d+81S3nDHVj
+dvpY6
  MDhoDcXm1Pt0Ne+/8TQ= true
```

Date: ……………………….

EXPERIMENT 13:

Aim: Study phishing in more detail. Find out which popular bank sites have been phished and how.

Phishing:

Phishing is the attempt to obtain sensitive information such as usernames, passwords, and credit card details (and, indirectly, money), often for malicious reasons, by disguising as a trustworthy entity in an electronic communication. The word is a neologism created as a homophone of *fishing* due to the similarity of using a bait in an attempt to catch a victim. According to the 2013 Microsoft Computing Safety Index, released in February 2014, the annual worldwide impact of phishing could be as high as US$5 billion.

Phishing is typically carried out by email spoofing[4] or instant messaging, and it often directs users to enter personal information at a fake website, the look and feel of which are identical to the legitimate one and the only difference is the URL of the website in concern. Communications purporting to be from social web sites, auction sites, banks, online payment processors or IT administrators are often used to lure victims. Phishing emails may contain links to websites that are infected with malware.

Phishing is an example of social engineering techniques used to deceive users, and exploits weaknesses in current web security. Attempts to deal with the growing number of reported phishing incidents include legislation, user training, public awareness, and technical security measures.

Phishing types:

- Spear phishing
- Clone phishing
- Whaling
- Link manipulation
- Filter evasion
- Website forgery

Date: ……………………….

- Covert redirect

- Social engineering
- Phone phishing

ICICI Bank Phishing:

A few customers of ICICI Bank received an e-mail asking for their Internet login name and password to their account. The e-mail seemed so genuine that some users even clicked on the URL given in the mail to a Web page that very closely resembled the official site. The scam was finally discovered when an assistant manager of ICICI Bank's information security cell received e-mails forwarded by the bank's customers seeking to crosscheck the validity of the e-mails with the bank. Such a scam is known as 'phishing.'

GERMAN Bank Phishing:

The CYREN team detected a massive phishing attack on customers of the German bank Postbank, with more than 50,000 new phishing URLs detected within the first 24 hours. Phishing emails are traditionally sent to a massive group of people, in the hope that among the recipients are actual customers of the brand and within that group there are unsuspecting users that will click the phishing link and complete whatever information request is included. In this case the email and phishing site look very similar to the legitimate Postbank website and so it is hard for regular users to see that this is actually phishing scam and unfortunately, enough people will fall victim to the scammer’s attempts.

Phishing attacks on banks and financial institutions are very common. In fact, the number one target of phishing scams is popular payment service PayPal. In the case of this Postbank scam the unsuspecting visitor is asked to confirm his login credentials. Once the user submits his username and password on the bogus website, the scammers have obtained the user’s credentials and are able to steal his identity or sell the information to another cybercriminal.

Date: ……………………….

EXTRA LAB EXPERIMENTS

EXPERIMENT 1:

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

PROGRAM:

```
#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
Hello World
```

Date: ……………………….

EXPERIMENT 2:

AIM: 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.

PROGRAM:

```
  #include <stdio.h>
  #include<stdlib.h>
  void main()
{
 char str[]="Hello World";
char str1[11];
char str2[11]=str[];
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++)
      {
      str3[i] = str2[i]^127;
   printf("%c",str3[i]);
           }
           printf("\n");
}
```

Date: ……………………….

OUTPUT:

```
Hello World
Hello World
Hello World
```

Date: ……………………….

EXPERIMENT 3:

AIM: Write a Java program to perform encryption and decryption using Ceaser Cipher Algorithm

PROGRAM:

```
importjava.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
 import java.util.Scanner
ublic 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.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)
```

Date: ……………………….

```
{
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;

           }

}
```

Date: ……………………….

OUTPUT:

```
 Enter any String: Hello World
Enter the Key: 5
Encrypted String is: MjqqtBtwqi
Decrypted String is: Hello World
```

Date: ……………………….

EXPERIMENT 4:

AIM: Write a Java program to perform encryption and decryption using Substitution Cipher algorithm.

PROGRAM:

```
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 {
    String a = "abcdefghijklmnopqrstuvwxyz";
    String b = "zyxwvutsrqponmlkjihgfedcba";

System.out.print("Enter any string: ");
    String str = br.readLine();
String decrypt = "";
char c;
for(int i=0;i<str.length();i++)
        {
 c = str.charAt(i); int j =a.indexOf(c);

decrypt = decrypt+b.charAt(j);
        }
System.out.println("The encrypted data is: " +decrypt);
   }
}
```

Date: ……………………….

OUTPUT:

```
 Enter any string: aceho
The encrypted data is: zxvsl
```

Date: ……………………….

EXPERIMENT 5:

AIM: Write a Java program to perform encryption and decryption using Hill Cipher algorithm.

PROGRAM:

```
 import java.io.*;
import java.util.*;
import java.io.*;
public class HillCipher {
static float[][] decrypt = new float[3][1]; static
float[][] a = new float[3][3];
static float[][] b = new float[3][3];
 static float[][] mes = new float[3][1];
static float[][] res = new float[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();
 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];
        }
inverse();
for(int i=0;i<3;i++)
```

Date: ……………………….

```
    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+97));
  }
    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.nextFloat();
    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 void inverse() {
    floatp,q; float[][] c = a;
     for(int i=0;i<3;i++)
    for(int j=0;j<3;j++) {
                //a[i][j]=sc.nextFloat(); if(i==j)
    b[i][j]=1; else b[i][j]=0;
            }
    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];
```

Date: ……………………….

```
                            } } } }
for(int i=0;i<3;i++)
for(int j=0;j<3;j++)
{ b[i][j] = b[i][j]/c[i][i];
 }
          System.out.println("");

          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.print("\n"); }

                 } }
```

OUTPUT:

```
 Enter a 3 letter string: hai

Encrypted string is :fdx

Inverse Matrix is:

0.083333336   0.41666666   -0.33333334

-0.41666666   -0.083333336   0.6666667

  0.5833333   -0.083333336   -0.33333334

Decrypted string is :hai
```