Problem Statement:

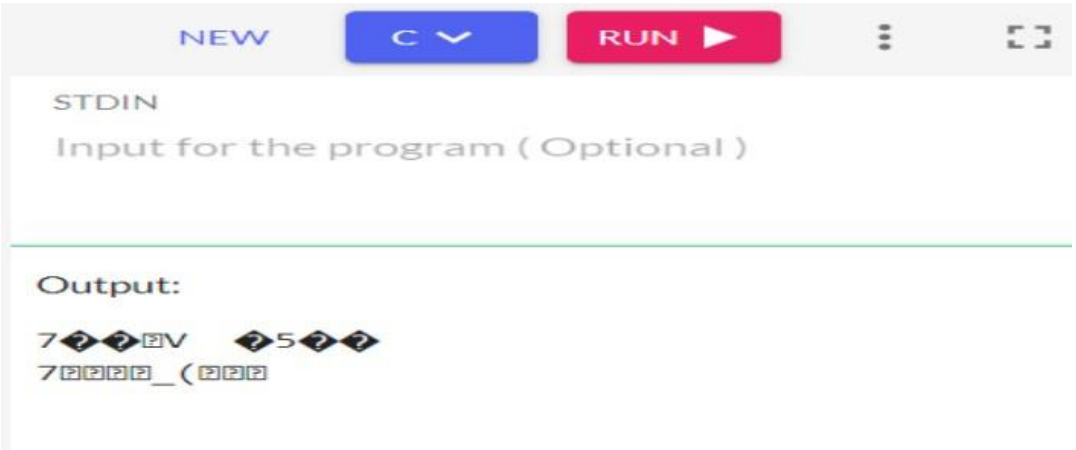
1)Write a Java/C/C++/Python program that contains a string (char pointer) with a value \Hello World'. The program should AND or and XOR each character in this string with 127 and display the result.

Code:

```
#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");
}
```

Output:



Problem Statement:

2) Write a Java/C/C++/Python program to implement DES algorithm.

Code:

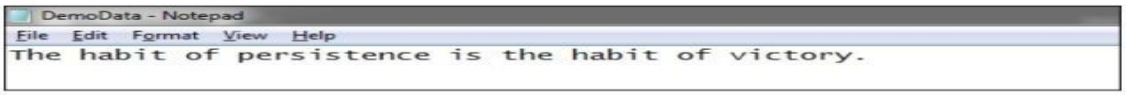
```
//Java classes that are mandatory to import for encryption and decryption process
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.AlgorithmParameterSpec;
import javax.crypto.Cipher;
import javax.crypto.CipherInputStream;
import javax.crypto.CipherOutputStream;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
public class DesProgram
{
//creating an instance of the Cipher class for encryption
private static Cipher encrypt;
//creating an instance of the Cipher class for decryption
private static Cipher decrypt;
//initializing vector
private static final byte[] initialization_vector = { 22, 33, 11, 44, 55, 99, 66, 77 };
//main() method
public static void main(String[] args)
{
//path of the file that we want to encrypt
String textFile = "C:/Users/Anubhav/Desktop/DemoData.txt";
//path of the encrypted file that we get as output
String encryptedData = "C:/Users/Anubhav/Desktop/encrypteddata.txt";
//path of the decrypted file that we get as output
```

```
String decryptedData = "C:/Users/Anubhav/Desktop/decrypteddata.txt";
try
{
//generating keys by using the KeyGenerator class
SecretKey scrtkey = KeyGenerator.getInstance("DES").generateKey();
AlgorithmParameterSpec aps = new IvParameterSpec(initialization_vector);
//setting encryption mode
encrypt = Cipher.getInstance("DES/CBC/PKCS5Padding");
encrypt.init(Cipher.ENCRYPT_MODE, scrtkey, aps);
//setting decryption mode
decrypt = Cipher.getInstance("DES/CBC/PKCS5Padding");
decrypt.init(Cipher.DECRYPT_MODE, scrtkey, aps);
//calling encrypt() method to encrypt the file
encryption(new FileInputStream(textFile), new FileOutputStream(encryptedData));
//calling decrypt() method to decrypt the file
decryption(new FileInputStream(encryptedData), new FileOutputStream(decryptedData));
//prints the stetment if the program runs successfully
System.out.println("The encrypted and decrypted files have been created successfully.");
}
//catching multiple exceptions by using the | (or) operator in a single catch block
catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException | Inval
idAlgorithmParameterException | IOException e)
{
//prints the message (if any) related to exceptions
e.printStackTrace();
}
}
//method for encryption
private static void encryption(InputStream input, OutputStream output)
throws IOException
{
output = new CipherOutputStream(output, encrypt);
//calling the writeBytes() method to write the encrypted bytes to the file
writeBytes(input, output);
}
//method for decryption
private static void decryption(InputStream input, OutputStream output)
```

```
throws IOException
{
input = new CipherInputStream(input, decrypt);
//calling the writeBytes() method to write the decrypted bytes to the file
writeBytes(input, output);
}
//method for writting bytes to the files
private static void writeBytes(InputStream input, OutputStream output)
throws IOException
{
byte[] writeBuffer = new byte[512];
int readBytes = 0;
while ((readBytes = input.read(writeBuffer)) >= 0)
{
output.write(writeBuffer, 0, readBytes);
}
//closing the output stream
output.close();
//closing the input stream
input.close();
}
}
```

Output:

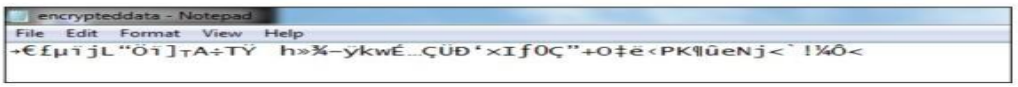
DemoData.txt



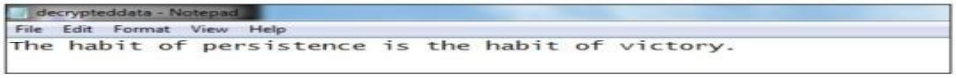
Let's run the above program and see the output.

Output:

encrypteddata.txt



deecrypteddata.txt



Problem Statement:

3) Write a Java/C/C++/Python program to implement AES Algorithm.

Code:

```
// Java program to demonstrate the creation
// of Encryption and Decryption with Java AES
import java.nio.charset.StandardCharsets;
import java.security.spec.KeySpec;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;

class AES {
    // Class private variables
    private static final String SECRET_KEY
        = "my_super_secret_key_ho_ho_ho";

    private static final String SALT = "ssshhhhhhhhhhh!!!!";

    // This method use to encrypt to string
    public static String encrypt(String strToEncrypt)
    {
        try {

            // Create default byte array
            byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0,
                          0, 0, 0, 0, 0, 0, 0, 0 };
            IvParameterSpec ivspec
                = new IvParameterSpec(iv);

            // Create SecretKeyFactory object
            SecretKeyFactory factory
                = SecretKeyFactory.getInstance(
                    "PBKDF2WithHmacSHA256");

            // Create KeySpec object and assign with
            // constructor
            KeySpec spec = new PBEKeySpec(
                SECRET_KEY.toCharArray(), SALT.getBytes(),
                65536, 256);
            SecretKey tmp = factory.generateSecret(spec);
            SecretKeySpec secretKey = new SecretKeySpec(
                tmp.getEncoded(), "AES");

            Cipher cipher = Cipher.getInstance(
```

```
            "AES/CBC/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey,
                ivspec);
        // Return encrypted string
        return Base64.getEncoder().encodeToString(
            cipher.doFinal(strToEncrypt.getBytes(
                StandardCharsets.UTF_8)));
    }
    catch (Exception e) {
        System.out.println("Error while encrypting: "
                    + e.toString());
    }
    return null;
}

// This method use to decrypt to string
public static String decrypt(String strToDecrypt)
{
    try {

        // Default byte array
        byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0 };
        // Create IvParameterSpec object and assign with
        // constructor
        IvParameterSpec ivspec
            = new IvParameterSpec(iv);

        // Create SecretKeyFactory Object
        SecretKeyFactory factory
            = SecretKeyFactory.getInstance(
                "PBKDF2WithHmacSHA256");

        // Create KeySpec object and assign with
        // constructor
        KeySpec spec = new PBEKeySpec(
            SECRET_KEY.toCharArray(), SALT.getBytes(),
            65536, 256);
        SecretKey tmp = factory.generateSecret(spec);
        SecretKeySpec secretKey = new SecretKeySpec(
            tmp.getEncoded(), "AES");

        Cipher cipher = Cipher.getInstance(
            "AES/CBC/PKCS5PADDING");
        cipher.init(Cipher.DECRYPT_MODE, secretKey,
                ivspec);
        // Return decrypted string
        return new String(cipher.doFinal(
            Base64.getDecoder().decode(strToDecrypt)));
    }
    catch (Exception e) {
        System.out.println("Error while decrypting: "
```

```
                        + e.toString());
        }
        return null;
    }
}

// driver code
public class Main {
    public static void main(String[] args)
    {
        // Create String variables
        String originalString = "GeeksforGeeks";

        // Call encryption method
        String encryptedString
            = AES.encrypt(originalString);

        // Call decryption method
        String decryptedString
            = AES.decrypt(encryptedString);

        // Print all strings
        System.out.println(originalString);
        System.out.println(encryptedString);
        System.out.println(decryptedString);
    }
}
```

Output:

Result

```
$javac Main.java

$java -Xmx128M -Xms16M Main

GeeksforGeeks
LuBu3DTLx7SLfjfhbjl7lw==
GeeksforGeeks
```

LP-II (IS part) 4

Problem Statement:

4)Write a Java/C/C++/Python program to implement RSA algorithm.

Code:

```
package com.sanfoundry.setandstring;

import java.io.DataInputStream;

import java.io.IOException;

import java.math.BigInteger;

import java.util.Random;

public class RSA

{

private BigInteger p;

private BigInteger q;

private BigInteger N;

private BigInteger phi;

private BigInteger e;

private BigInteger d;

private int bitlength = 1024;

private Random r;

public RSA()

{

r = new Random();

p = BigInteger.probablePrime(bitlength, r);

q = BigInteger.probablePrime(bitlength, r);

N = p.multiply(q);

phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));

e = BigInteger.probablePrime(bitlength / 2, r);

while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0)

{

e.add(BigInteger.ONE);

}
```

```
d = e.modInverse(phi);
}
public RSA(BigInteger e, BigInteger d, BigInteger N)
{
this.e = e;
this.d = d;
this.N = N;
}
@SuppressWarnings("deprecation")
public static void main(String[] args) throws IOException
{
RSA rsa = new RSA();
DataInputStream in = new DataInputStream(System.in);
String teststring;
System.out.println("Enter the plain text:");
teststring = in.readLine();
System.out.println("Encrypting String: " + teststring);
System.out.println("String in Bytes: "
+ bytesToString(teststring.getBytes()));
// encrypt
byte[] encrypted = rsa.encrypt(teststring.getBytes());
// decrypt
byte[] decrypted = rsa.decrypt(encrypted);
System.out.println("Decrypting Bytes: " + bytesToString(decrypted));
System.out.println("Decrypted String: " + new String(decrypted));
}
private static String bytesToString(byte[] encrypted)
{
String test = "";
for (byte b : encrypted)
{
```

```
test += Byte.toString(b);
}
return test;
}
// Encrypt message
public byte[] encrypt(byte[] message)
{
return (new BigInteger(message)).modPow(e, N).toByteArray();
}
// Decrypt message
public byte[] decrypt(byte[] message)
{
return (new BigInteger(message)).modPow(d, N).toByteArray();
}
}
```

Output:

```
$ javac RSA.java
$ java RSA

Enter the plain text:
Sanfoundry
Encrypting String: Sanfoundry
String in Bytes: 83971101021111171101001141 21
Decrypting Bytes: 83971101021111171101001141 21
Decrypted String: Sanfoundry
```

LP-II (IS part)

Problem Statement:

5) 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).

Code:

```
// This program calculates the Key for two persons
// using the Diffie-Hellman Key exchange algorithm
class MGF{

// Power function to return value of a ^ b mod P
private static long power(long a, long b, long p)
{
	if (b == 1)
		return a;
	else
		return (((long)Math.pow(a, b)) % p);
}

// Driver code
public static void main(String[] args)
{
	long P, G, x, a, y, b, ka, kb;

	// Both the persons will be agreed upon the
	// public keys G and P

	// A prime number P is taken
	P = 23;
```

```
        System.out.println("The value of P:" + P);

        // A primitive root for P, G is taken
        G = 9;
        System.out.println("The value of G:" + G);

        // Alice will choose the private key a
        // a is the chosen private key
        a = 4;
        System.out.println("The private key a for Alice:" + a);

        // Gets the generated key
        x = power(G, a, P);

        // Bob will choose the private key b
        // b is the chosen private key
        b = 3;
        System.out.println("The private key b for Bob:" + b);

        // Gets the generated key
        y = power(G, b, P);

        // Generating the secret key after the exchange
        // of keys
        ka = power(y, a, P); // Secret key for Alice
        kb = power(x, b, P); // Secret key for Bob

        System.out.println("Secret key for the Alice is:" + ka);
        System.out.println("Secret key for the Bob is:" + kb);
    }
}
```

Output:

The value of P : 23

The value of G : 9

The private key a for Alice : 4

The private key b for Bob : 3

Secret key for the Alice is : 9

Secret Key for the Bob is : 9