



GR22 Regulations
Cryptography and Network Security Lab
(GR22A4055)

Department of Computer Science and Engineering

GOKARAJU RANGARAJU
INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Autonomous)



Gokaraju Rangaraju Institute of Engineering and Technology (Autonomous)

Vision and Mission of the Institution

Vision:

To be among the best of the institutions for engineers and technologists with attitudes, skills and knowledge and to become an epicenter of creative solutions.

Mission:

To achieve and impart quality education with an emphasis on practical skills and social relevance.

Vision and Mission of the Department

Vision of the Department

To be a center of global excellence and to emerge as a valuable resource for industry and society.

Mission of the Department

The Computer Science and Engineering is committed to

1. Produce qualified and competent computer professionals with international standards.
2. Foster innovative and application oriented research capabilities of young minds for the progress of society.
3. Inculcate strong ethical values and professional behavior so as to adapt to the emerging changes in the field of computing technology.



**Gokaraju Rangaraju Institute of Engineering and Technology
(Autonomous)**

Bachupally, Kukatpally, Hyderabad – 500 090

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAMME EDUCATIONAL OBJECTIVES(PEOs)

PEO 1: Graduates will be prepared for a successful career in Computer Science discipline and related industry to meet the needs of the nation and leading industries and also to excel in postgraduate programs.

PEO 2: Graduates will continue to learn and apply the acquired knowledge to solve engineering problems and appreciation of the arts, humanities and social sciences.

PEO 3: Graduates will have good and broad scientific and engineering knowledgebase so as to comprehend, analyze, design and create novel products and solutions for real-time applications.

PEO 4: Graduates will understand professional and ethical responsibility, develop leadership, utilize membership opportunities, develop effective communication skills, teamwork skills, multidisciplinary approach and life-long learning required for a successful professional career.

PROGRAM OUTCOMES (POs)

PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems.

PO2. Problem analysis: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. Design/Development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.

PO4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and the need for sustainable development.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. Communication: Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions

PO11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES(PSOs)

PSO1: Emerging Technologies: Apply the concepts of Computer Science and Engineering to learn the emerging technologies and to develop inventive solutions.

PSO2: Industry oriented technical skills: To foster industrial focused technical skills in Computer science and Engineering through value added courses and, to create a futuristic equipped professional.

**GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND
TECHNOLOGY**
CRYPTOGRAPHY AND NETWORK SECURITY LAB

Course Code: GR22A4055

L/T/P/C: 0/0/4/2

Course Outcomes:

1. Interpret the ciphers used for encryption and decryption.
2. Implement the symmetric encryption algorithms.
3. Analyze the asymmetric encryption algorithms.
4. Summarize the hash algorithms and classes related to digital certificates.
5. Illustrate the intrusion detection and web security systems.

List of Tasks:

TASK 1: Write a Java program to perform encryption and decryption using the following algorithms.

a. Ceaser cipher b. Substitution cipher c. Hill Cipher

TASK 2: Implement symmetric block cipher encryption and decryption using DES algorithm in C/JAVA.

TASK 3: Write a C/JAVA program to implement encryption technique using Blowfish algorithm.

TASK 4: Implement the encryption of block chunk of 128 bits size using AES algorithm in C/JAVA.

TASK 5: Write a C/JAVA program on Rivest Cipher 4(RC4) logic.

TASK 6: Implement DES-2 and DES-3 using Java cryptography package.

TASK 7: Design a Java program to implement RSA algorithm.

TASK 8: Implement key exchange protocol using the Diffie-Hellman algorithm.

TASK 9: Calculate the message digest of a text using the SHA-1 algorithm in JAVA.

Task 10: Calculate the message digest of a text using the MD5 algorithm in JAVA.

Task 11: Explore the Java classes related to digital certificates.

Task 12: Implement a program in java, which performs Cross-site scripting(XSS) Attacks

INDEX

S.No	Tasks	Page No.
1	Write a Java program to perform encryption and decryption using the following algorithms. a. Ceaser cipher b. Substitution cipher c. Hill Cipher	1
2	Implement symmetric block cipher encryption and decryption using DES algorithm in C/JAVA.	10
3	Write a C/JAVA program to implement encryption technique using Blowfish algorithm.	12
4	Implement the encryption of block chunk of 128 bits size using AES algorithm in C/JAVA.	14
5	Write a C/JAVA program on Rivest Cipher 4(RC4) logic.	16
6	Implement DES-2 and DES-3 using Java cryptography package.	19
7	Design a Java program to implement RSA algorithm.	21
8	Implement key exchange protocol using the Diffie-Hellman algorithm.	23
9	Calculate the message digest of a text using the SHA-1 algorithm in JAVA.	25
10	Calculate the message digest of a text using the MD5 algorithm in JAVA.	27
11	Explore the Java classes related to digital certificates.	29
12	Implement a program in java, which performs Cross-site scripting(XSS) Attacks.	31

TASK 1

Task 1(a):Write a Java program to perform encryption and decryption using the algorithm Ceaser Cipher.

Aim:To write a Program on encryption and decryption using ceaser cipher.

Description: To encrypt a message with a ceaser cipher,the letter in each message is changed using a simple rule : shift by three. Each letter is replaced by the letter three letters ahead in the alphabet .A becomes D,B becomes E,and so on.for the last letters , we can think of the alphabet as a circle and “wrap around”.W becomes Z,X becomes A,Y becomes B and Z becomes C.To change a message back, each letter is replaced by the one three before it.

Program:

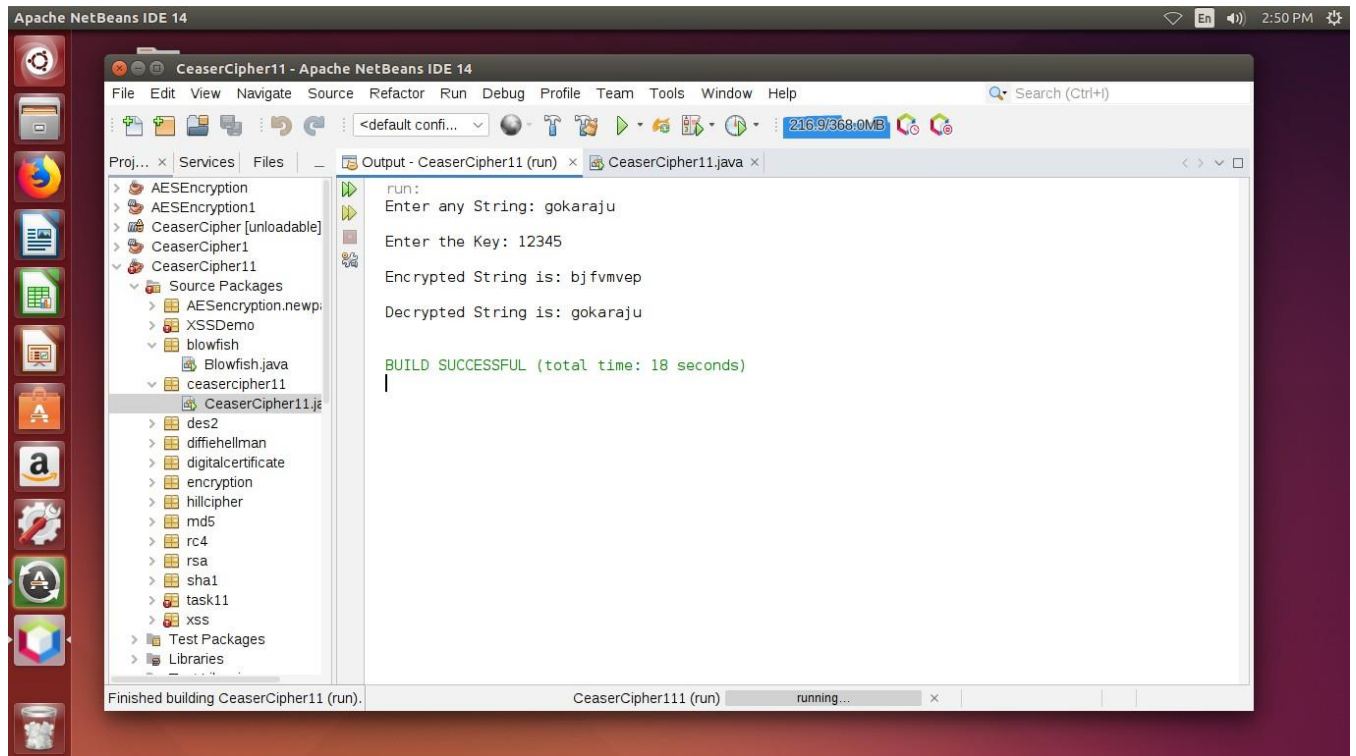
```
import java.security.*;
import java.io.*;
import java.util.*;
//Ceaser Cipher
public class Task1a {
    static Scanner s = new Scanner(System.in);

    private static String encrypt(String data, int key) {
        StringBuilder res = new StringBuilder();
        for (Character ch :
            data.toCharArray()) { if
            (Character.isUpperCase(ch)) {
                char shiftedChar = (char) (((ch - 'A' + key) % 26) +
                    'A'); res.append(shiftedChar);
            } else if (Character.isLowerCase(ch)) {
                char shiftedChar = (char) (((ch - 'a' + key) % 26) +
                    'a'); res.append(shiftedChar);
            } else {
                res.append(ch);
            }
        }
        return res.toString();
    }

    private static String decrypt(String data, int key)
    { return encrypt(data, 26 - (key % 26));
    }

    public static void main(String[] args) {
        String data = s.nextLine();
        int key = s.nextInt();
        data = encrypt(data, key);// Encryption
        System.out.println("Encryption: " + data);
        System.out.println("Decryption: " + decrypt(data, key));// Decryption
    }
}
```

OUTPUT:



VIVA QUESTIONS

1. How do you decrypt a message that was encrypted using the Caesar Cipher?
2. What are the limitations of the Caesar Cipher in terms of security?
3. How can the Caesar Cipher be broken using brute force or frequency analysis?
4. What is the fundamental principle behind the Caesar Cipher?

Task 1(b):Write a Java program to perform encryption and decryption using the algorithm Substitution Cipher.

Aim: To write a Java Program using substitution cipher.

Description:

This program demonstrates how to perform **encryption and decryption using the Substitution Cipher technique** in Java. The Substitution Cipher is a classical encryption method in which each letter of the plaintext is replaced with another letter based on a fixed mapping (key). For example, if 'A' is mapped to 'D', then every 'A' in the message is substituted with 'D'.

Program :

```
package com.islab;

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 encrypt = encrypt();
        String decrypt = decrypt(encrypt);
        System.out.println("The encrypted data is: " +encrypt);
        System.out.println("The decrypted data is: " +decrypt);
    }

    public static String encrypt()throws IOException
    {
        String encrypt = "";
        String a = "abcdefghijklmnopqrstuvwxyz";
        String b = "zyxwvutsrqponmlkjihgfedcba";
        System.out.print("Enter any string: ");
        String str = br.readLine();
        char c;
        for(int i=0;i<str.length();i++)
        {
            c = str.charAt(i);
            int j = a.indexOf(c);
            encrypt = encrypt+b.charAt(j);
        }
    }
}
```

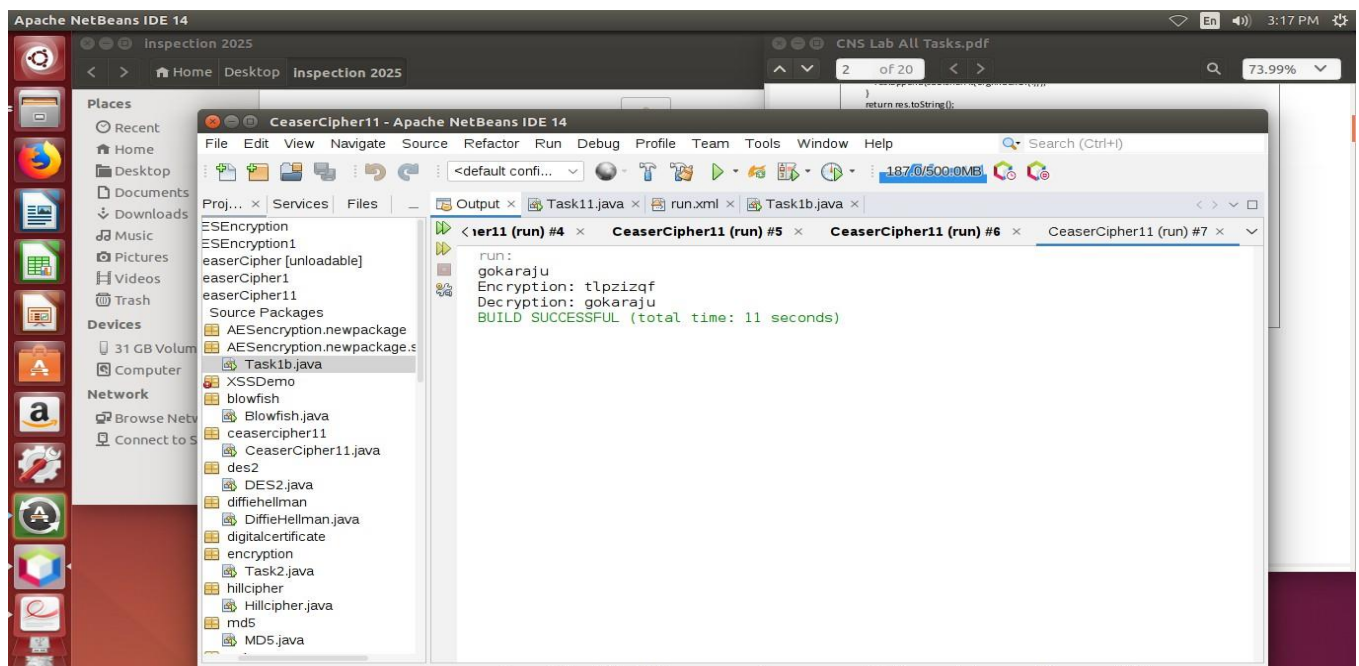
```
    return encrypt;  
}
```

```

public static String decrypt(String encrypt){
    String a = "abcdefghijklmnopqrstuvwxyz";
    String b = "zyxwvutsrqponmlkjihgfedcba";
    String decrypt = "";
    char c;
    for(int i=0;i<encrypt.length();i++) {
        c = encrypt.charAt(i);
        int j = a.indexOf(c);
        decrypt = decrypt+b.charAt(j);
    }
    return decrypt;
}
}

```

OUTPUT:



VIVA QUESTIONS

1. What is the Substitution Cipher, and how does it differ from the Caesar Cipher?
2. What are the different types of substitution ciphers? Provide examples.
3. How can a Substitution Cipher be broken using frequency analysis?
4. What are the advantages and disadvantages of using a Substitution Cipher?

Task 1(c):Write a Java program to perform encryption and decryption using the algorithm Hill Cipher.

Aim:To write a Java Program using Hill cipher.

Description:

Each letter is represented by a number modulo 26. Often the simple scheme $A=0, B=1 \dots Z=25$, as used, but this is not an essential feature of the cipher. To encrypt a message, each block of n letters is multiplied by an invertible $n \times n$ matrix, against modulus 26. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption. The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible $n \times n$ matrices (modulo 26).

Program:

```
package com.islab;
import java.util.*;
class Basic
{
    String allChar="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    int indexOfChar(char c)
    {
        for(int i=0;i < allChar.length();i++)
        {
            if(allChar.charAt(i)==c)
                return i;
        }
        return -1;
    }

    char charAtIndex(int pos)
    {
        return allChar.charAt(pos);
    }
}
class Hill
{
    Hill(int block)
    {
        this.block=block;
    }

    Basic b1=new Basic();
    int block=2;
    int key[][]=new int[block][block];

    void keyInsert()throws Exception
```

```
{  
Scanner scn=new Scanner(System.in);  
System.out.println("Enter key Matrix");  
    for(int i=0;i < block;i++)  
    {  
        for(int j=0;j < block;j++)
```

```

        {
            key[i][j]=scn.nextInt();
        }
    }
}

```

void KeyInverseInsert()throws Exception

```

{
    Scanner scn=new Scanner(System.in);
    System.out.println("Enter key Inverse Matrix:");
    for(int i=0;i < block;i++)
    {
        for(int j=0;j < block;j++)
        {
            key[i][j]=scn.nextInt();
        }
    }
}

```

String encryptBlock(String plain)throws Exception

```

{
    plain=plain.toUpperCase();
    int a[][]=new int[block][1],sum=0;
    int cipherMatrix[][]=new int[block][1];
    String cipher="";
    for(int i=0;i < block;i++)
    {
        a[i][0]=b1.indexOfChar(plain.charAt(i));
    }
    for(int i=0;i < block;i++)
    {
        for(int j=0;j < 1;j++)
        {
            for(int k=0;k < block;k++)
            {
                sum=sum+key[i][k]*a[k][j];
            }
            cipherMatrix[i][j] = sum%26;
            sum = 0;
        }
    }

    for(int i=0;i < block;i++)
    {
        cipher+=b1.charAtIndex(cipherMatrix[i][0]);
    }
}

```



```

        }
    return cipher;
}
String encrypt(String plainText)throws Exception
{
    String cipherText="";
    keyInsert();
    plainText=plainText.toUpperCase();
    int len=plainText.length();
    // System.out.println(plainText.substring(1,2+1));
    while(len%block!=0)
    {
        plainText+="X";
        System.out.println(len);
        len=plainText.length();
    }
    for(int i=0;i < len-1;i=i+block)
    {
        cipherText+=encryptBlock(plainText.substring(i,i+block));
        cipherText+=" ";
    }
    return cipherText;
}

String decryptBl(String cipher)throws Exception
{
    cipher=cipher.toUpperCase();
    int a[][]=new int[block][1],sum=0;
    int plainMatrix[][]=new int[block][1];
    String plain="";
    for(int i=0;i < block;i++)
    {
        a[i][0]=b1.indexOfChar(cipher.charAt(i));
    }
    for(int i=0;i < block;i++)
    {
        for(int j=0;j < 1;j++)
        {
            for(int k=0;k < block;k++)
            {
                sum=sum+key[i][k]*a[k][j];
            }
            while(sum < 0)
            {
                sum+=26;
            }
        }
    }
    plain+=a[i][0];
}

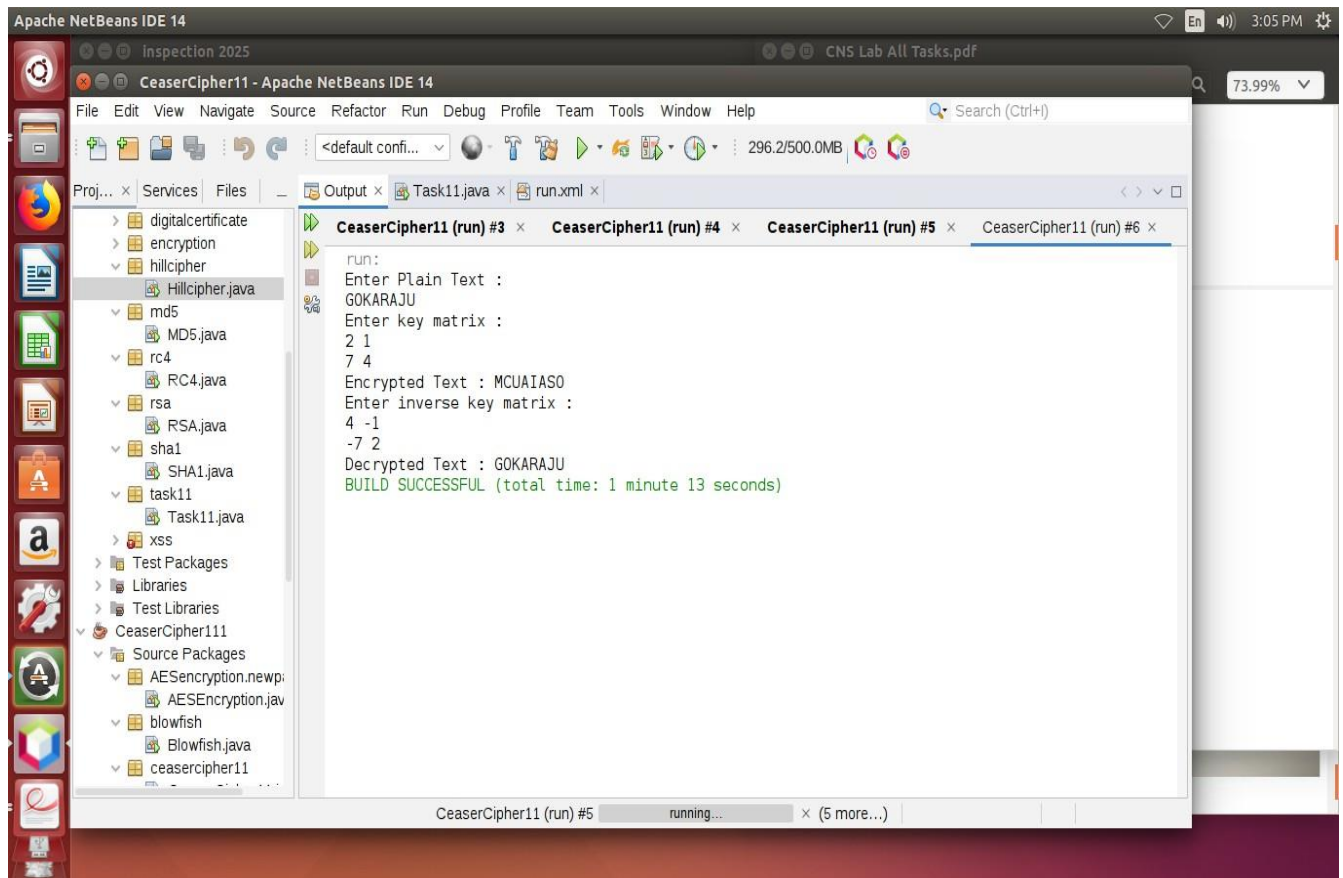
```

```

        }
        plainMatrix[i][j] = sum;
        sum = 0;
    }
    orf(int i=0;i < block;i++)
    {
        plain+=b1.charAtIndex(plainMatrix[i][0]);
    }
    return plain;
)
String Decrypt(String cipherText)throws Exception
{
    String plainText="";
    KeyInverseInsert();
    cipherText=cipherText.replaceAll(" ", "");
    cipherText=cipherText.toUpperCase();
    int len=cipherText.length();
    for(int i=0;i < len-1;i=i+block)
    {
        plainText+=decryptBlock(cipherText.substring(i,i+block));
        plainText+=" ";
    }
    return plainText;
}
}
class HillCipher
{
    public static void main(String args[])throws Exception
    {
        String plainText,cipherText;
        int block;
        Scanner scn=new Scanner(System.in);
        System.out.println("Enter plain-text:");
        plainText=scn.nextLine();
        System.out.println("Enter block size of matrix:");
        block=scn.nextInt();
        Hill hill=new Hill(block);
        plainText=plainText.replaceAll(" ", "");
        cipherText= hill.encrypt(plainText);
        System.out.println("Encrypted Text is:\n"+cipherText);
        String decryptedText= hill.Decrypt(cipherText);
        System.out.println("Decrypted Text is:\n"+decryptedText);
    }
}

```

OUTPUT:



VIVA QUESTIONS

1. Why does the key matrix in Hill Cipher need to be invertible in modular arithmetic?
2. How do you perform encryption and decryption using the Hill Cipher?
3. What are the advantages and limitations of the Hill Cipher compared to other classical ciphers?
4. How does matrix multiplication play a role in the Hill Cipher encryption process?

TASK 2

Implement symmetric block cipher encryption and decryption using DES algorithm in C/JAVA

Aim: To write a Java Program on DES algorithm

Description:

The program demonstrates the implementation of symmetric block cipher encryption and decryption using the **Data Encryption Standard (DES) algorithm** in Java. DES is a symmetric key algorithm, meaning the same secret key is used for both encryption and decryption. The program accepts plaintext and a secret key as input, encrypts the plaintext into ciphertext using DES, and then decrypts the ciphertext back into the original plaintext.

This implementation showcases the working of the DES algorithm, highlighting the process of secure data transmission using a shared key. It helps in understanding how block ciphers operate in symmetric cryptography, ensuring confidentiality of data during communication.

Program:

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

//DES
public class Task2 {
    static Scanner s = new
Scanner(System.in); private static Cipher
cipher;
    private static byte[] encrypt(String data) throws Exception{
        byte[] encrypted= cipher.doFinal(data.getBytes());
        System.out.println(encrypted);
        return encrypted;
    }
    private static byte[] decrypt(byte[] data) throws Exception{
        byte[] decrypted= cipher.doFinal(data);
        System.out.println(new
String(decrypted)); return decrypted;
    }
    public static void main(String[] args)
    { try {
```

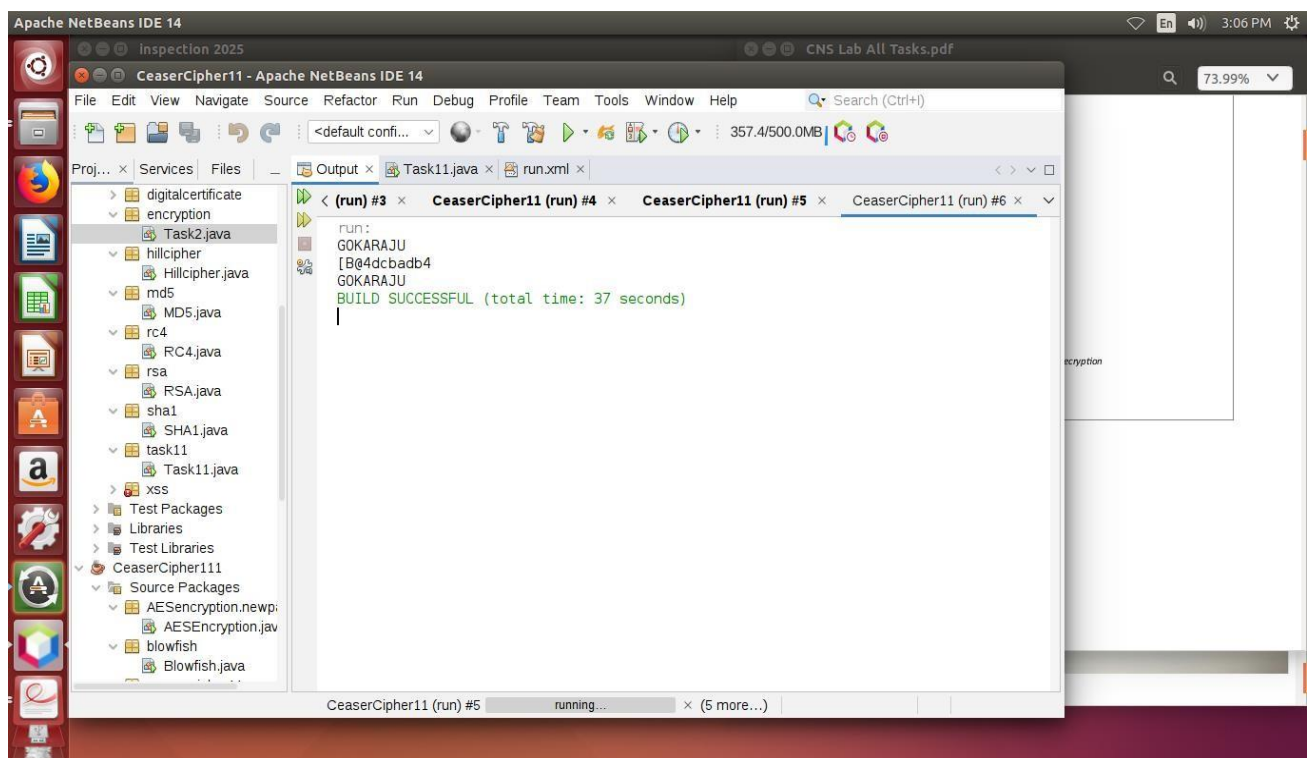
```
String data = s.nextLine();  
KeyGenerator keygen =  
KeyGenerator.getInstance("DES");  
SecretKey key = keygen.generateKey();  
cipher=Cipher.getInstance("DES/ECB/PKCS5Paddin");  
cipher.init(Cipher.ENCRYPT_MODE,key);  
byte[] encrypted = encrypt(data);
```

```

        cipher.init(Cipher.DECRYPT_MODE, key);
        decrypt(encrypted);
    } catch (Exception e)
    {
        e.printStackTrace();
    }
}
}
}

```

OUTPUT:



VIVA QUESTIONS

1. What is a block cipher, and how does it differ from a stream cipher?
2. Can you explain the working of a common block cipher, such as AES or DES?
3. What are the different modes of operation used in block ciphers, and why are they important?
4. How does key length impact the security of a block cipher?

TASK 3

Write a C/JAVA program to implement encryption technique using Blowfish algorithm

Aim: To write a Program to implement Blowfish algorithm logic.

Description:

This program demonstrates the implementation of an encryption technique using the **Blowfish algorithm**. Blowfish is a symmetric block cipher designed by Bruce Schneier, which operates on 64-bit blocks and uses variable-length keys (from 32 bits up to 448 bits). It is known for its speed, simplicity, and strong security.

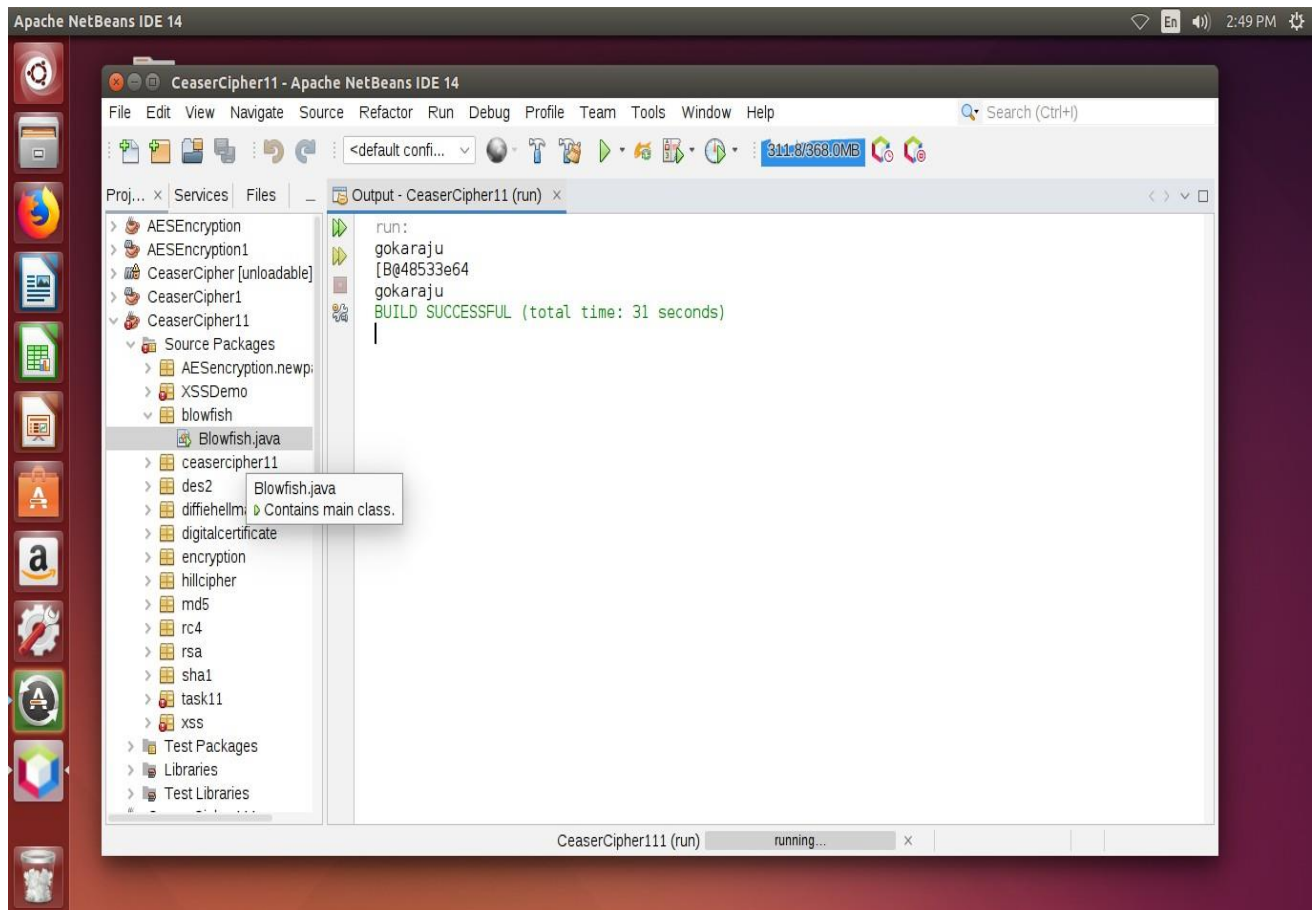
Program:

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

//DES
public class Task3 {
    static Scanner s = new
    Scanner(System.in); private static Cipher
    cipher;
    private static byte[] encrypt(String data) throws Exception{
        byte[] encrypted= cipher.doFinal(data.getBytes());
        System.out.println(encrypted);
        return encrypted;
    }
    private static byte[] decrypt(byte[] data) throws Exception{
        byte[] decrypted= cipher.doFinal(data);
        System.out.println(new
        String(decrypted)); return decrypted;
    }
    public static void main(String[] args)
    {
        try
        {
            String data = s.nextLine();
            KeyGenerator keygen = KeyGenerator.getInstance("Blowfish");
            SecretKey key = keygen.generateKey();
            cipher=Cipher.getInstance("Blowfish"
            );
            cipher.init(Cipher.ENCRYPT_MODE
            ,key); byte[] encrypted =
            encrypt(data);
            cipher.init(Cipher.DECRYPT_MODE,key);
            decrypt(encrypted);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

}
}

OUTPUT:



VIVA QUESTION

1. How does the Blowfish encryption process work, and what makes it secure?
2. What are the advantages and disadvantages of using Blowfish compared to AES and DES?
3. How does the Blowfish key expansion process contribute to its security?
4. Why is Blowfish not recommended for modern applications despite its strong security features?

TASK 4

Implement the encryption of block chunk of 128 bits size using AES algorithm in C/JAVA

Aim: To implement AES Algorithm logic.

Description:

This program implements the **Advanced Encryption Standard (AES)** algorithm to perform encryption on a **block of data with size 128 bits (16 bytes)**. AES is a symmetric key block cipher standardized by NIST, widely used for secure data transmission.

The program takes a plaintext block of 128 bits and a secret key as input, then applies the AES encryption process. The steps include:

1. **Key Expansion** – The given key is expanded into multiple round keys using Rijndael's key schedule.
2. **Initial Round** – AddRoundKey operation (XORing plaintext with the first round key).
3. **Main Rounds** (9 rounds for AES-128):
 - **SubBytes:** Non-linear substitution using the S-box.
 - **ShiftRows:** Cyclic shifting of rows in the state matrix.
 - **MixColumns:** Mixing of data within columns using matrix multiplication in $GF(2^8)$.
 - **AddRoundKey:** XOR of the state with the round key.
4. **Final Round** (10th round for AES-128):
 - SubBytes
 - ShiftRows
 - AddRoundKey (without MixColumns).

The output is a **ciphertext block (128 bits)**, which represents the encrypted form of the input plaintext.

Program:

```
import java.security.*;
import java.io.*;

import javax.crypto.*;
import java.util.*;

public class Task4 {

    static Scanner s = new
Scanner(System.in); private static Cipher
cipher;

    private static byte[] encrypt(String data) throws Exception{
        byte[] encrypted= cipher.doFinal(data.getBytes());

        System.out.println(encrypted);
        return encrypted;
    }
}
```

```

private static byte[] decrypt(byte[] data) throws Exception{
    byte[] decrypted= cipher.doFinal(data);

    System.out.println(new
    String(decrypted)); return decrypted;
}

public static void main(String[] args)
{ try {
    String data = s.nextLine();

    KeyGenerator keygen = KeyGenerator.getInstance("AES");
    SecretKey key = keygen.generateKey();

    cipher=Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE
    ,key); byte[] encrypted =
    encrypt(data);

    cipher.init(Cipher.DECRYPT_MODE,key);
    decrypt(encrypted);
} catch (Exception e)

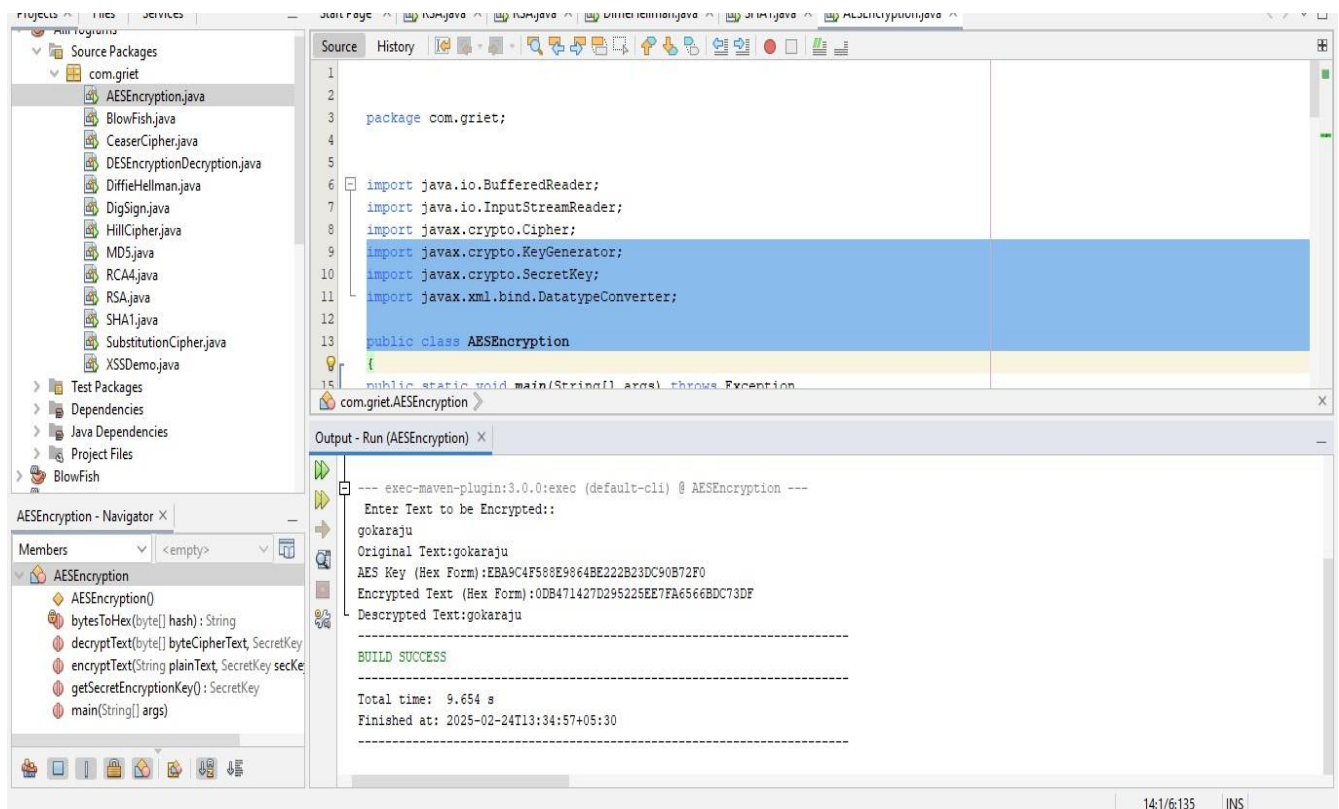
```

```

    {
        e.printStackTrace();
    }
}
}
}

```

OUTPUT:



VIVA QUESTIONS

1. What is AES, and why was it chosen as the standard encryption algorithm?
2. How does AES perform encryption and decryption, and what are its key components?
3. What are the different key sizes used in AES, and how do they affect security?
4. How does AES resist cryptographic attacks like brute force and differential cryptanalysis?

TASK 5

Write a C/JAVA program on Rivest Cipher 4(RC4) logic.

Aim:To implement RC4 LOGIC

Description:

The program is designed to implement the **Rivest Cipher 4 (RC4)** logic, which is a widely used symmetric stream cipher. RC4 works by generating a pseudorandom keystream that is combined with the plaintext (for encryption) or ciphertext (for decryption) using the bitwise XOR operation.

The implementation includes the following steps:

1. **Key Scheduling Algorithm (KSA):**
 - Initializes a permutation of all 256 possible byte values (S-box).
 - Uses the secret key to shuffle the S-box.
2. **Pseudo-Random Generation Algorithm (PRGA):**
 - Produces a sequence of pseudorandom bytes based on the shuffled S-box.
 - Each byte from this keystream is XORed with the input text to produce encrypted or decrypted output.
3. **Encryption/Decryption Process:**
 - Since RC4 is a symmetric cipher, the same algorithm is applied for both encryption and decryption.
 - Encrypted text can be decrypted by applying RC4 again with the same key.

Program:

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

public class Task5 {

    static Scanner s = new Scanner(System.in);

    private static String encrypt(String data, String key)
    {
        int[] s = new int[256];

        int[] k = new
        int[256];

        int temp, i, j = 0, z;

        StringBuilder encryptedData = new StringBuilder();

        char[] dataChars =
        data.toCharArray(); char[]
        keyChars = key.toCharArray();
        int[] dataInt = new
```

```
int[data.length()]; int[] keyInt =  
new int[key.length()];  
for (int m = 0; m < data.length(); m++)  
{  
    dataInt[m] = (int) dataChars[m];  
}  
for (int m = 0; m < key.length(); m++)  
{  
    keyInt[m] = (int) keyChars[m];  
}  
for (int m = 0; m < 256; m++)  
{
```

```

        s[m] = m;
        k[m] = keyInt[m % key.length()];
    }

    j = 0;
    for (int m = 0; m < 256; m++)
    {
        j = (j + s[m] + k[m]) % 256;
        temp = s[m];
        s[m] = s[j];
        s[j] = temp;
    }

    i = 0;
    j = 0;
    for (int l = 0; l < data.length(); l++)
    {
        i = (i + 1) % 256;
        j = (j + s[i]) % 256;

        temp =
            s[i]; s[i] =
            s[j]; s[j] =
            temp;

        z = s[(s[i] + s[j]) % 256];
        encryptedData.append((char) (z ^ dataInt[l]));
    }

    return encryptedData.toString();
}

private static String decrypt(String data, String
    key) { return encrypt(data, key);
}

public static void main(String[] args) {
    System.out.println("Enter the plain
text:"); String data = s.nextLine();
    System.out.println("Enter the key:");

```

```

String key = s.nextLine();

String encryptedData = encrypt(data, key);

System.out.println("Encrypted: " + encryptedData);

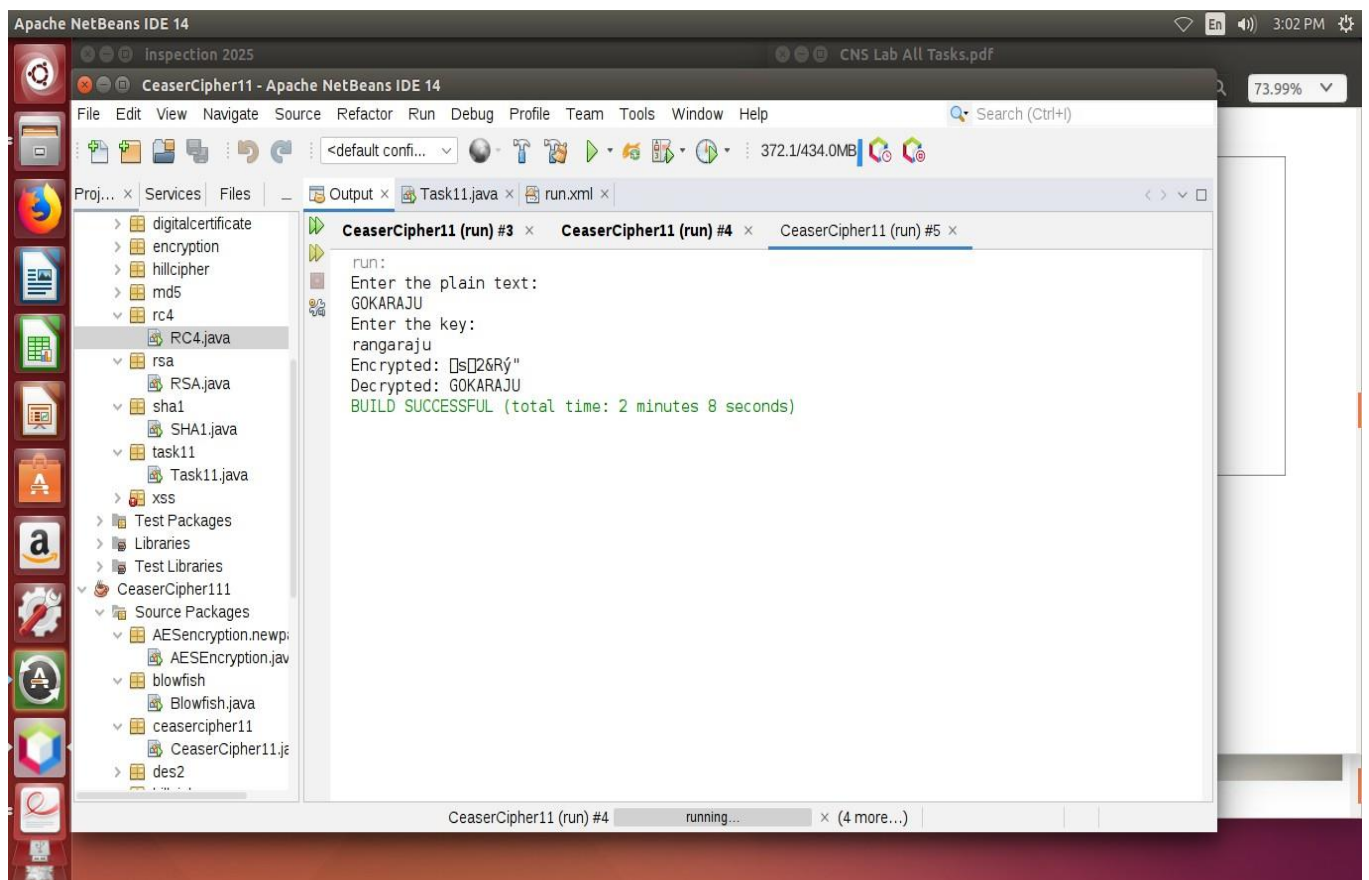
System.out.println("Decrypted: " + decrypt(encryptedData, key));

}

}

```

OUTPUT:



VIVA QUESTIONS

1. What is the RC4 algorithm, and how does it work?
2. What is the role of the key-scheduling algorithm (KSA) in RC4?
3. What are the major vulnerabilities of RC4, and why is it considered insecure today?
4. Where was RC4 commonly used, and what algorithms have replaced it in modern cryptography?

TASK 6

Implement DES-2 and DES-3 using Java cryptography package

Aim: To implement DES-2 and DES-3 using Java cryptography package

Description:

The objective of this program is to demonstrate the implementation of **Double DES (DES-2)** and **Triple DES (DES-3)** encryption techniques using the **Java Cryptography Architecture (JCA)**. These algorithms extend the standard DES encryption to provide enhanced security by applying the DES algorithm multiple times with different keys.

- **Double DES (DES-2):**

In DES-2, the plaintext is first encrypted using one DES key and then decrypted using a second DES key, effectively increasing the key space and security compared to single DES. The process can be represented as:

`Ciphertext = E(K2, D(K1, Plaintext))`

- **Triple DES (DES-3):**

In DES-3 (also called **3DES** or **Triple Data Encryption Standard**), the plaintext undergoes **three DES operations** with either two or three keys. A common form is:

`Ciphertext = E(K3, D(K2, E(K1, Plaintext)))`

This process ensures backward compatibility with single DES while significantly improving cryptographic strength.

- **Java Cryptography Package Usage:**

The program utilizes classes from the `javax.crypto` package such as:

- `Cipher` (for encryption and decryption)
- `SecretKey` and `SecretKeyFactory` (for managing keys)
- `KeyGenerator` (for generating DES keys)

Program:

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

public class Task6 {
    static Scanner s = new Scanner(System.in);
    private static Cipher cipher;
    private static byte[] encrypt(String data, SecretKey key1, SecretKey key2, int flag) throws
Exception {
        String algo = (flag == 2) ? "DESede" : "DES";
        cipher = Cipher.getInstance(algo +
"/ECB/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, key1);
        byte[] encrypted = cipher.doFinal(data.getBytes());

        if (flag == 1) {
            cipher.init(Cipher.ENCRYPT_MODE,
key2); encrypted =
cipher.doFinal(encrypted);
        }
        System.out.println("Encrypted: " + new
```

```

        String(encrypted)); return encrypted;
    }
    private static String decrypt(byte[] data, SecretKey key1, SecretKey key2, int flag) throws
Exception {
    String algo = (flag == 2) ? "DESede" :
    "DES"; byte[] decrypted;
    cipher = Cipher.getInstance(algo + "/ECB/PKCS5Padding");
    if (flag == 1) {
        cipher.init(Cipher.DECRYPT_MODE,
        key2); decrypted =
        cipher.doFinal(data);
    } else {
        decrypted = data;
    }
    cipher.init(Cipher.DECRYPT_MODE,
    key1); decrypted =
    cipher.doFinal(decrypted);
    System.out.println("Decrypted: " + new String(decrypted));
    return new String(decrypted);
}
public static void main(String[]
args) { try {

```

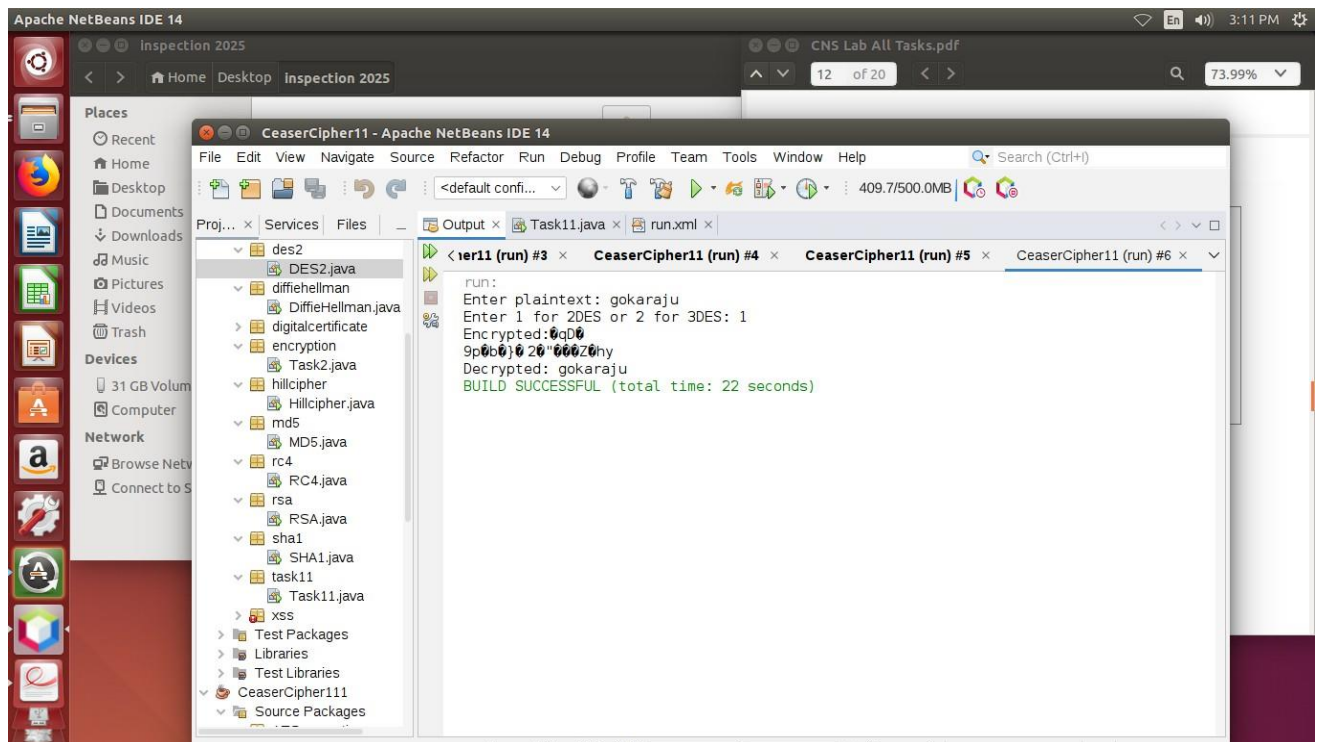
```
System.out.print("Enter plaintext: "); String data = s.nextLine();
```

```

    System.out.print("Enter 1 for 2DES or 2 for
    3DES: "); int flag = s.nextInt();
    KeyGenerator keygen = KeyGenerator.getInstance((flag == 2) ? "DESede" :
    "DES"); SecretKey key1 = keygen.generateKey();
    SecretKey key2 = keygen.generateKey();
    byte[] encrypted = encrypt(data, key1, key2, flag);
    decrypt(encrypted, key1, key2, flag);
} catch (Exception e)
{
    e.printStackTrace();
}
}
}

```

OUTPUT:



VIVA QUESTIONS

1. What is the main difference between DES, Double DES (DES-2), and Triple DES (DES-3)?
2. Why is Double DES (DES-2) vulnerable to the meet-in-the-middle attack?
3. How does Triple DES (DES-3) improve security over DES-2?
4. What are the different modes of operation for Triple DES (DES-3)?

TASK 7

Write a JAVA Program to implement the RSA algorithm.

Aim: To implement RSA Algorithm.

Description:

The RSA algorithm is one of the most widely used **public-key cryptographic techniques**. It relies on the mathematical difficulty of factoring large prime numbers. In this program, two large prime numbers are selected to generate the modulus (n) and the totient function (ϕ). A public key (e) is chosen such that it is relatively prime to ϕ , and a corresponding private key (d) is computed as the modular inverse of e modulo ϕ .

Program:

```
import java.math.*;
import java.util.*;

//RSA

public class Task7 {
    static Scanner s = new Scanner(System.in);
    private static BigInteger p, q, N, phi, e, d;
    private static int bitlength = 1024;
    private static Random r;
    private static void Setter() {
        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 = e.add(BigInteger.ONE);
        }
        d = e.modInverse(phi);
    }
    private static byte[] encrypt(String data) {
        return (new BigInteger(data.getBytes())).modPow(e, N).toByteArray();
    }
    private static byte[] decrypt(byte[] encryptedData) {
        return (new BigInteger(encryptedData)).modPow(d, N).toByteArray();
    }
}
```

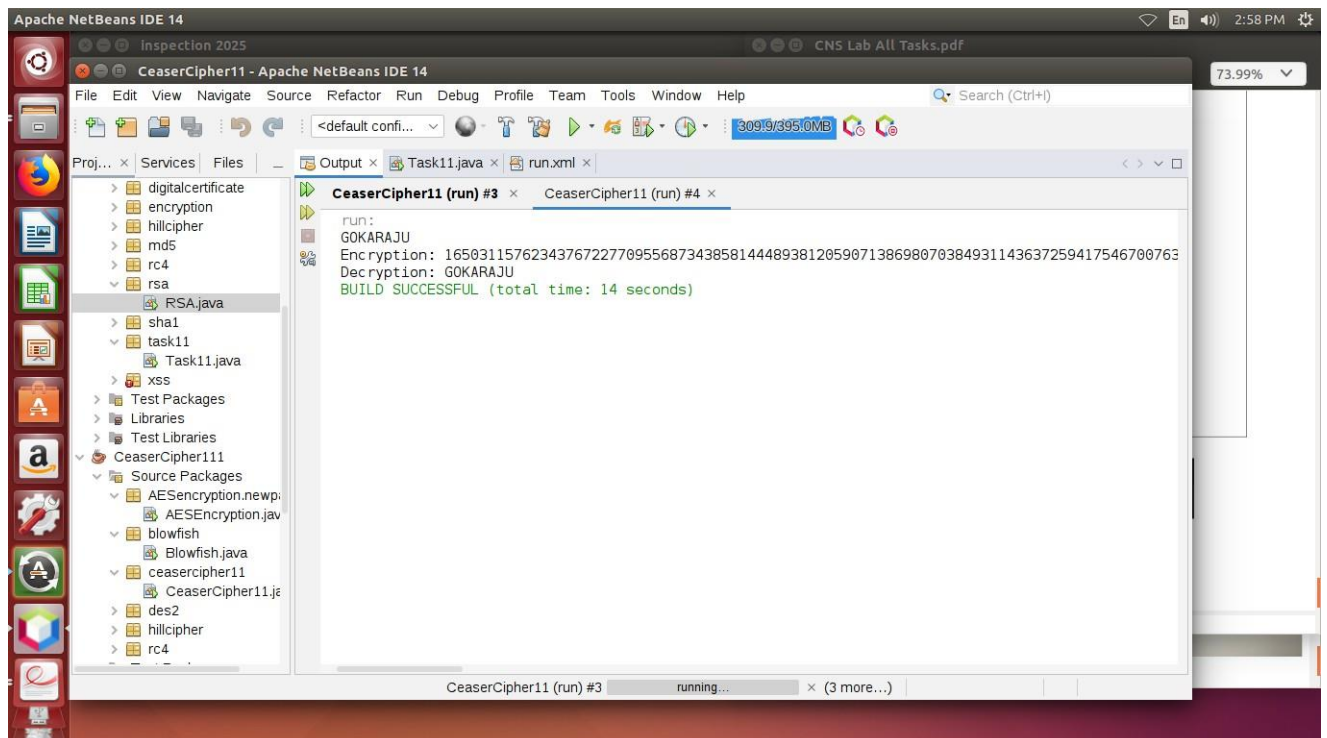
```
}  
public static void main(String[] args) { String data = s.nextLine();  
    Setter();  
    // Encryption
```

```

byte[] encrypted = encrypt(data);
System.out.println("Encryption: " + new BigInteger(encrypted).toString());
// Decryption
System.out.println("Decryption: " + new String(decrypt(encrypted)));
}
}

```

Output:



VIVA QUESTIONS

1. What is the RSA algorithm, and how does it work?
2. How are the public and private keys generated in RSA?
3. What are the main advantages and disadvantages of using RSA for encryption?
4. How does RSA encryption compare to symmetric encryption algorithms like AES in terms of performance and security?

TASK 8

Implement key exchange protocol using the Diffie-Hellman algorithm

Aim: To implement the Diffie-Hellman Key Exchange mechanism.

Program Description

The program demonstrates the working of the **Diffie–Hellman Key Exchange protocol**, which is one of the earliest practical methods for securely exchanging cryptographic keys over a public channel.

The aim of this implementation is to show how two parties (commonly called **Alice** and **Bob**) can agree upon a shared secret key without directly transmitting the key itself. The protocol relies on the mathematical difficulty of solving the **discrete logarithm problem**.

Program:

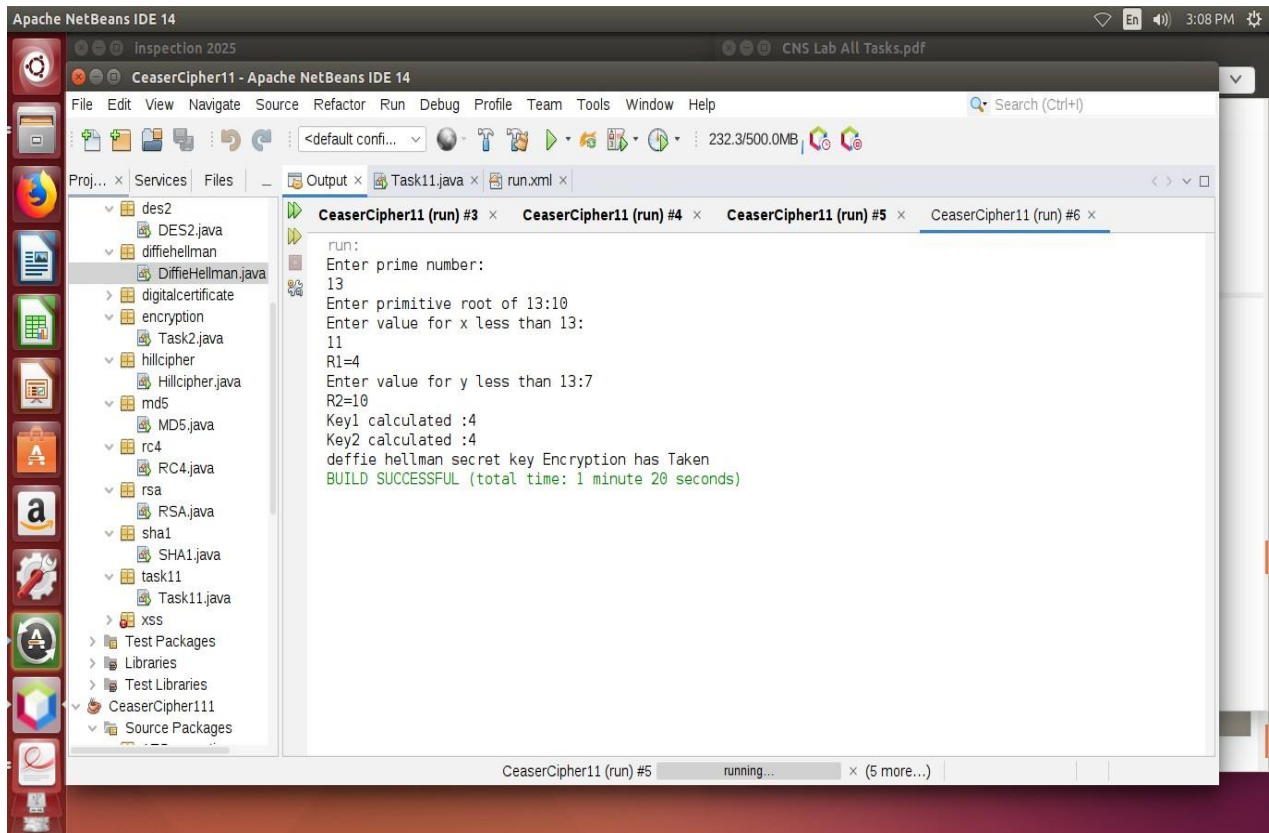
```
package com.islab;

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

public class DiffieHellman
{
    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("Key1 calculated :"+k1);
        BigInteger k2=R1.modPow(y,p);
        System.out.println("Key2 calculated :"+k2);
    }
}
```

```
        System.out.println("deffie hellman secret key Encryption has Taken");  
    }  
}
```


OUTPUT:



```
run:
Enter prime number:
13
Enter primitive root of 13:10
Enter value for x less than 13:
11
R1=4
Enter value for y less than 13:7
R2=10
Key1 calculated :4
Key2 calculated :4
diffie hellman secret key Encryption has Taken
BUILD SUCCESSFUL (total time: 1 minute 20 seconds)
```

VIVA QUESTIONS

1. What is the Diffie-Hellman algorithm, and what problem does it solve?
2. How does the Diffie-Hellman key exchange process work?
3. Why is the Diffie-Hellman algorithm considered secure, and what mathematical concept underlies its security?
4. What are the main vulnerabilities of Diffie-Hellman, and how can they be mitigated?

TASK 9

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

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

Description:

This program demonstrates how to calculate the **message digest** (hash value) of a given text using the **SHA-1 (Secure Hash Algorithm 1)** in Java. The SHA-1 algorithm produces a fixed-length 160-bit (20-byte) hash value, which is commonly represented as a 40-digit hexadecimal number.

In the program, the user provides an input string (text). The Java `MessageDigest` class, from the `java.security` package, is used to apply the SHA-1 algorithm to this text. The resulting hash (digest) is then converted into a readable hexadecimal format.

This ensures that even a small change in the input text generates a completely different digest, which makes SHA-1 useful for verifying data integrity and detecting modifications in transmitted or stored information.

Program:

```
package com.islab;

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

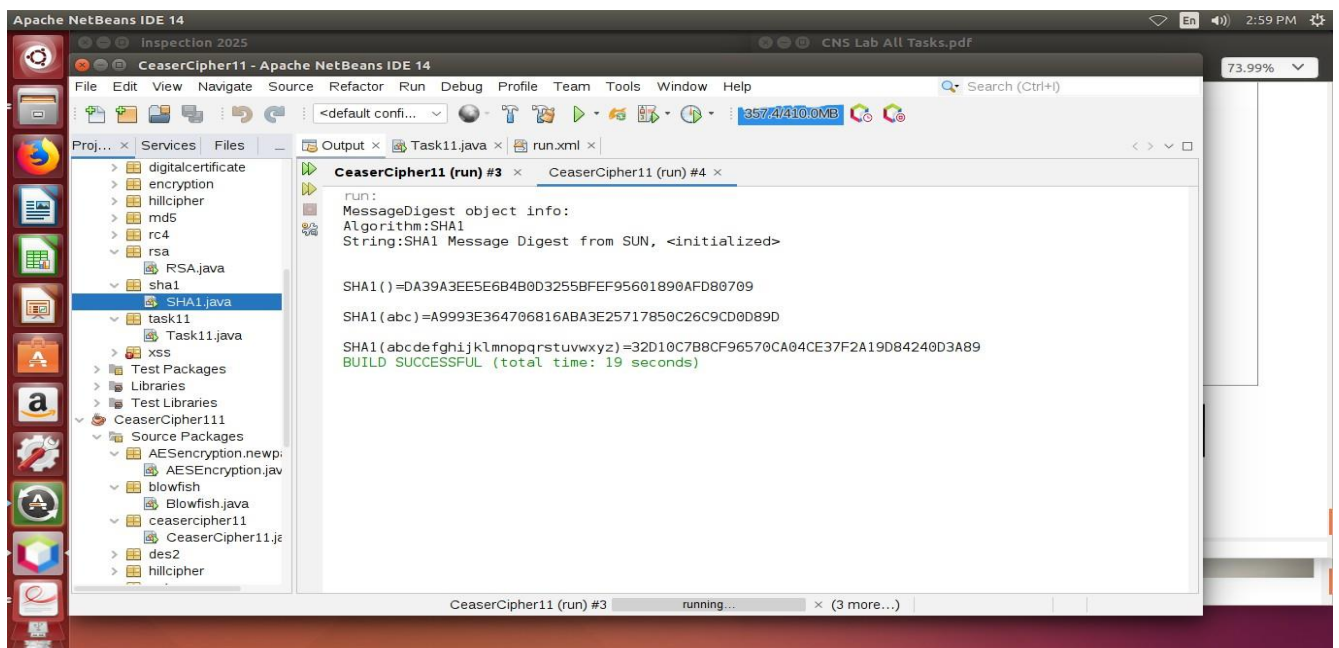
```

        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:



VIVA QUESTIONS

1. What is the length of the SHA-1 hash output?
2. What is the block size used in SHA-1?
3. How many rounds does SHA-1 use in its hashing process?
4. Explain the working steps of the SHA-1 algorithm

TASK 10

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

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

Description:

The program demonstrates how to generate the **message digest** (a fixed-size hash value) of a given text using the **MD5 algorithm** in Java. MD5 (Message Digest Algorithm 5) is a widely used cryptographic hash function that takes an input (text) and produces a 128-bit hash value, typically represented as a 32-character hexadecimal number.

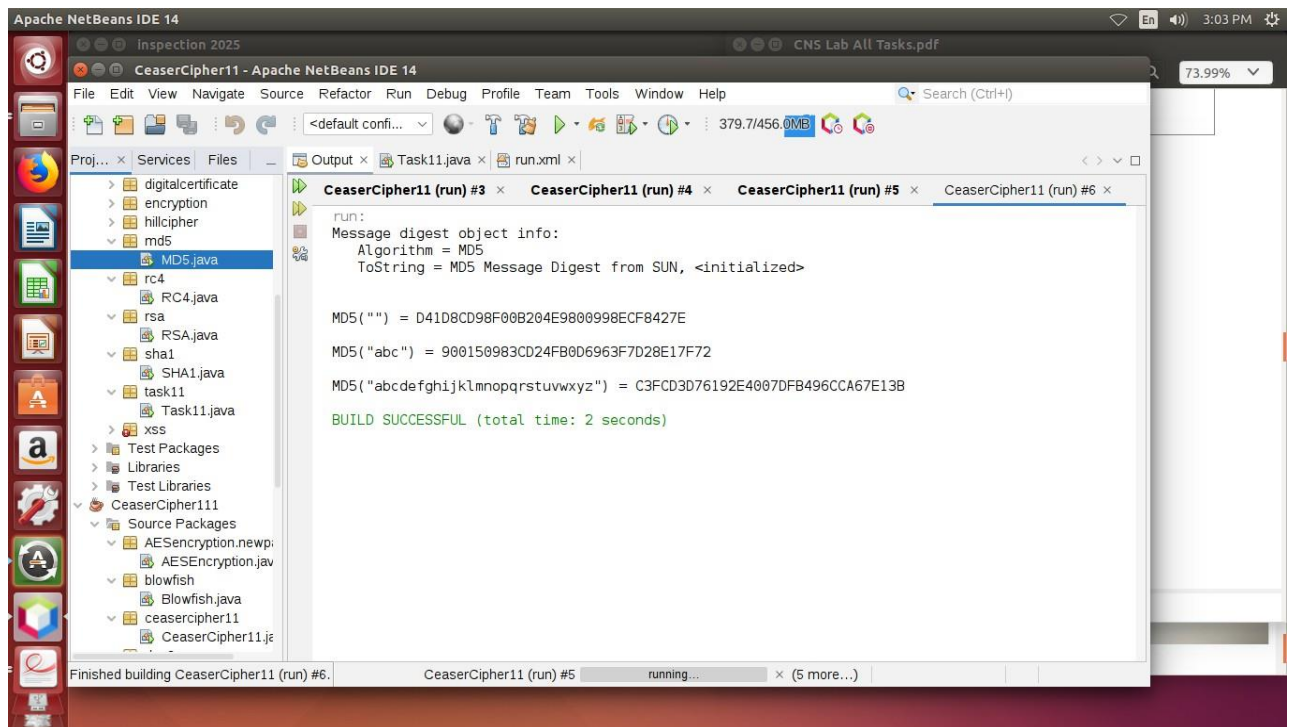
In this program, Java's built-in **MessageDigest** class from the `java.security` package is used. The input string is first converted into a byte array, then passed to the MD5 algorithm to compute the hash. Finally, the resulting byte array is transformed into its hexadecimal representation and displayed as the **message digest** of the given text.

Program:

```
import java.security.*;
import java.util.*;
public class Task10 {
    static Scanner s = new
Scanner(System.in); public static void
main(String[] args) {
    try {
        System.out.print("Enterinputdata:");
        String data = s.nextLine();

        MessageDigest md = MessageDigest.getInstance("MD5");
        System.out.println("Message digest object info: ");
        System.out.println(" Algorithm = " +
md.getAlgorithm());
        System.out.println("ToString="+
md.toString());md.update(data.getBytes());
        System.out.println("MD5(\"" + data + "\")="+HexFormat.of().formatHex(md.digest()));
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
}
```

OUTPUT:



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

MD5("") = D41D8CD98F00B204E9800998ECF8427E
MD5("abc") = 900150983CD24FB0D6963F7D28E17F72
MD5("abcdefghijklmnopqrstuvwxyz") = C3FCD3D76192E4007DFB496CCA67E13B

BUILD SUCCESSFUL (total time: 2 seconds)
```

VIVA QUESTIONS

1. How many rounds does MD5 use in its hashing process?
2. Explain the step-by-step working of the MD5 algorithm.
3. What are the four main operations used in MD5's compression function?
4. What is the length of the MD5 hash output?

TASK 11

Explore the java classes related to Digital Certificates.

Aim: To explore the Java classes related to Digital Certificates.

Description:

This program focuses on examining the Java security framework, particularly the classes under the `java.security.cert` package, which provide mechanisms for creating, managing, and validating digital certificates. Digital certificates are widely used to establish trust, authenticate entities, and ensure secure communication in distributed systems.

Program:

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

public class Task11 {
    private static Scanner s = new Scanner(System.in);
    public static void main(String[] args) {
        try {

            // Generating key pair (public and private keys)
            KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DSA");
            keyGen.initialize(1024);
            KeyPair keyPair = keyGen.generateKeyPair();
            PrivateKey privateKey = keyPair.getPrivate();
            PublicKey publicKey = keyPair.getPublic();

            // Sample data to be signed
            String data = s.nextLine();
            byte[] dataBytes = data.getBytes();

            // Signing the data
            Signature signature = Signature.getInstance("SHA256withDSA");
            signature.initSign(privateKey); // Initialize for signing
            signature.update(dataBytes); // Add data
            byte[] digitalSignature = signature.sign(); // Generate the digital signature

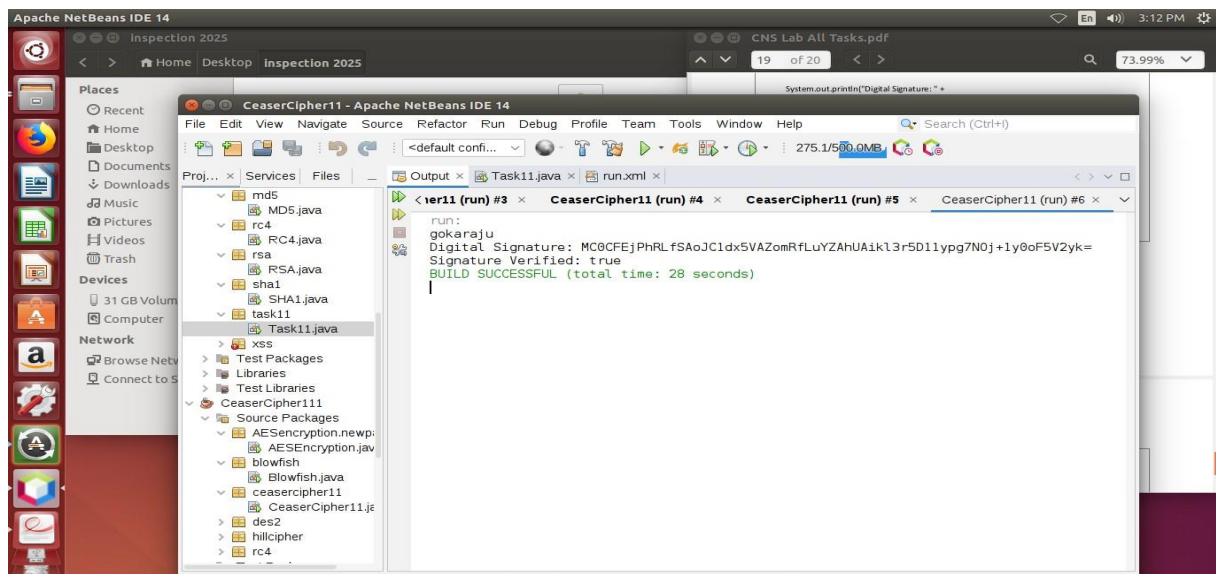
            System.out.println("DigitalSignature:" + Base64.getEncoder().encodeToString(digitalSignature));

            // Verifying the signature
            signature.initVerify(publicKey); // Initialize for verification
            signature.update(dataBytes); // Add data to be verified
            boolean isVerified = signature.verify(digitalSignature); // Verify the signature

            System.out.println("Signature Verified: " + isVerified);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

}
}
}

OUTPUT



VIVA QUESTIONS

1. What is a **Public Key Infrastructure (PKI)**?
2. How does a digital certificate verify the authenticity of a user or website?
3. What are the key components of a digital certificate?
4. What is the difference between a self-signed certificate and a CA-issued certificate?

TASK 12

Implement a program in java, which performs Cross-site scripting(XSS) Attacks

Aim:To write a program which performs Cross-site scripting(XSS) Attacks.

Description:

This program is designed for educational purposes to simulate a **Cross-Site Scripting (XSS) attack** in a controlled environment. XSS is a common web security vulnerability that allows attackers to inject malicious scripts into trusted websites. The injected script executes in the browser of unsuspecting users, leading to theft of cookies, session hijacking, redirection to malicious sites, or unauthorized actions.

In this demonstration, a simple **Java-based web application** is used to show how user input, when not properly validated or sanitized, can be exploited. The program accepts user data (e.g., through an input field or query parameter) and displays it directly on a webpage without applying proper security filters. By entering script tags or JavaScript code, a user can trigger an XSS payload.

Program:

```
import java.util.scanner;

import org.apache.commons.text.StringEscapeUtils;//for preventing xss

public class XSSDemo{

    public static void main(string[ ] args){

        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter your comment");

        String userInput= scanner.nextLine();

        System.out.println("without xss protection user inputs is..." +userinput);

        String safeinput= StringEscapeUtils.escapeHtml4(userinput);

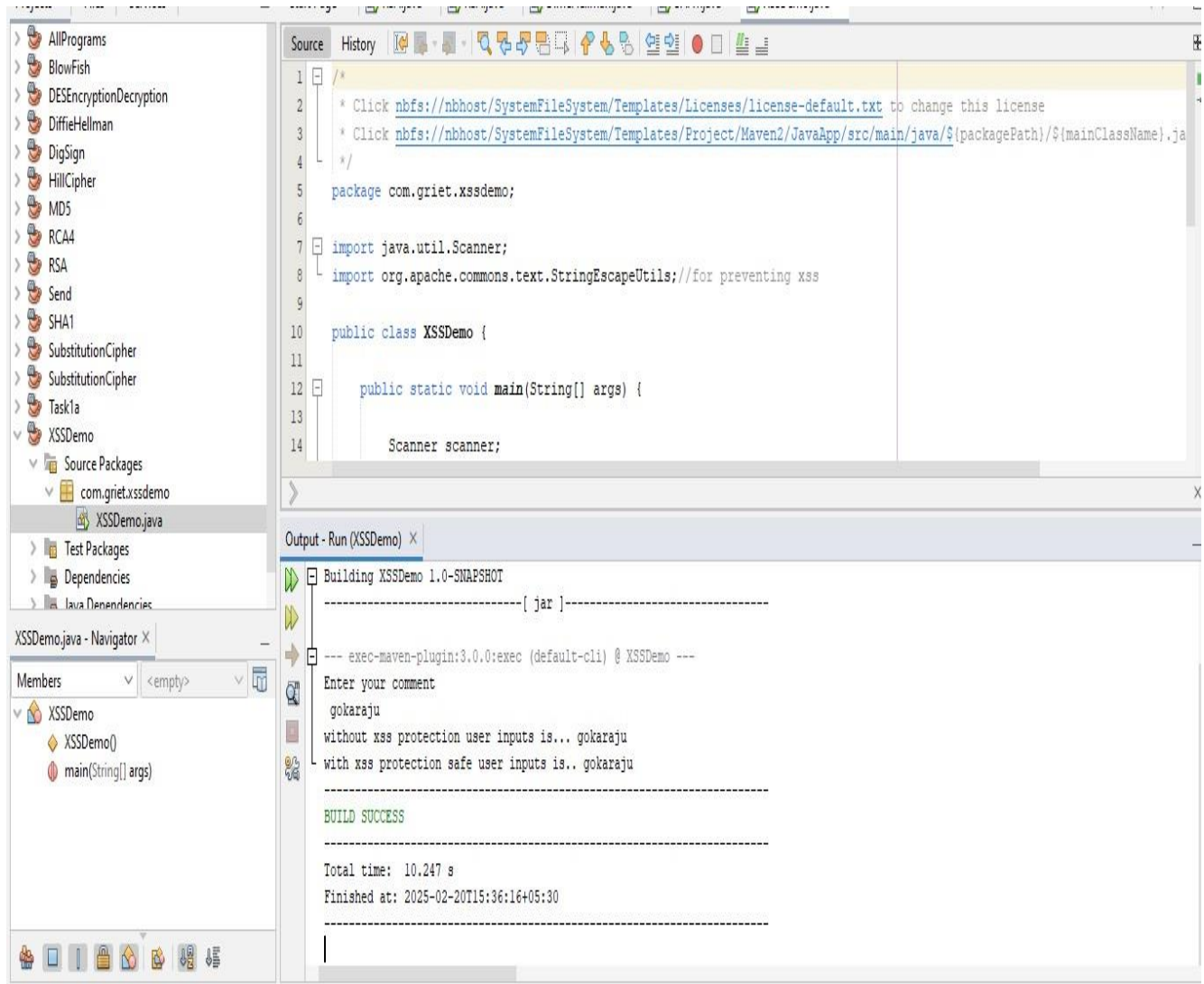
        System.out.println("with xss protection safe user inputs is.." +safeinput);

        scanner.close();

    }

}
```

OUTPUT



VIVA QUESTIONS

1. What is **Stored XSS**, and how does it work?
2. What is **Reflected XSS**, and how does it differ from stored XSS?
3. What is **DOM-based XSS**, and how is it different from the other types?
4. Which type of XSS attack is the most dangerous, and why?