

# DEPARTMENT OF COMPUTER ENGINEERING & APPLICATIONS

**Institute of Engineering & Technology**

**Cryptography & Network Security Lab (BCSE-0071)**

### Name: Anvesh Mangalam

### Section: H (17)

### University Roll No.: 201500133

### Course: B.Tech CSE

### Submitted to: Mr.Asheesh Tiwari Sir

INDEX

|  |  |  |
| --- | --- | --- |
| **Serial**  **No.** | **Title** | **Sign.** |
| **1.** | Write a program to implement Additive Cipher (Z26) with the following conditions: Plaintext should be in lowercase, Cipher text should be uppercase,  Brute force attack |  |
| **2.** | Write a program to implement Multiplicative Cipher: Plaintext should be in lowercase, Cipher text should be uppercase, Brute force attack. |  |
| **3.** | Write a program to implement Affine Cipher: Plaintext should be in lowercase, Cipher text should be uppercase, Brute force attack. |  |
| **4.** | Write a program in to implement Auto key Cipher: Plaintext should be in lowercase, Cipher text should be uppercase, Brute force attack. |  |
| **5.** | Write a program to implement Play fair Cipher to encrypt & decrypt the given message where the key matrix can be formed by using a given keyword. |  |
| **6.** | Write a program to implement Hill Cipher to encrypt & decrypt the given message by using a given key matrix. Show the values for key and its corresponding key inverse values. |  |
| **7.** | Write a program to implement Elgamal Cryptosystem to generate the pair of keys and then show the encryption & decryption of a given message. |  |
| **8.** | Write a program to implement Rabin Miller Primality Test to check whether given number is prime or composite. |  |
| **9.** | Write a program to implement Diffie-Hellman key exchange Algorithm to exchange the symmetric key and show the encryption & decryption. |  |
| **10.** | Write a program to implement RSA Algorithm to generate a pair of keys and show the encryption and decryption by using a given key pair. |  |
| **11.** | Write a program to implement Elgamal algorithm for implementing digital signature. |  |

# Experiment 1

Write a program to implement Additive Cipher (Z26) with the following conditions:

* Plaintext should be in lowercase.
* Ciphertext should be uppercase.
* Brute force attack.

# Source Code:

Objective – Write the Algorithm for Additive Cipher in Cryptography.

Code –

import java.util.InputMismatchException;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.println("\nWelcome to the Additive Cipher Program!");

int choice = 0;

do {

System.out.println("\nSelect an option:");

System.out.println("1. Encryption");

System.out.println("2. Decryption");

System.out.println("3. Brute Force");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

try {

choice = input.nextInt();

input.nextLine(); // Consume the newline character

switch (choice) {

case 1:

encryption();

break;

case 2:

decryption();

break;

case 3:

bruteForce();

break;

case 4:

System.out.println("\nGoodbye!");

break;

default:

System.out.println("\nInvalid choice. Please choose from 1-4.");

break;

}

} catch (InputMismatchException e) {

System.out.println("\nInvalid input. Please enter a valid choice (1-4).");

input.nextLine(); // Consume the invalid input

}

} while (choice != 4);

input.close();

}

public static void encryption() {

Scanner input = new Scanner(System.in);

System.out.println("\nYou selected Encryption.");

System.out.print("Enter the plaintext: ");

String plainText = input.nextLine();

if (!plainText.matches("[a-z ]+")) {

System.out.println("Error: The plaintext should only contain lower letters.");

return;

}

plainText = plainText.toUpperCase(); // Convert input to uppercase

int key = getValidKey(input);

String cipherText = "";

for (int i = 0; i < plainText.length(); i++) {

char ch = plainText.charAt(i);

ch = (char) ((ch - 'A' + key) % 26 + 'A');

cipherText += ch;

}

System.out.println("The ciphertext is: " + cipherText

)

}

public static void decryption() {

Scanner input = new Scanner(System.in);

System.out.println("\nYou selected Decryption.");

System.out.print("Enter the ciphertext: ");

String cipherText = input.nextLine();

// Check if the input contains non-uppercase letters

if (!cipherText.matches("[A-Z ]+")) {

System.out.println("Error: The ciphertext should only contain uppercase letters.");

return;

}

cipherText = cipherText.toUpperCase(); // Convert input to uppercase

int key = getValidKey(input);

String plainText = "";

for (int i = 0; i < cipherText.length(); i++) {

char ch = cipherText.charAt(i);

ch = (char) ((ch - 'A' - key + 26) % 26 + 'A');

plainText += ch;

}

System.out.println("The plaintext is: " + plainText);

}

public static void bruteForce() {

Scanner input = new Scanner(System.in);

System.out.println("\nYou selected Brute Force.");

System.out.print("Enter the ciphertext: ");

String cipherText = input.nextLine();

if (!cipherText.matches("[A-Z ]+")) {

System.out.println("Error: The ciphertext should only contain uppercase letters.");

return;

}

cipherText = cipherText.toUpperCase(); // Convert input to uppercase

for (int key = 0; key < 26; key++) {

String plainText = "";

for (int i = 0; i < cipherText.length(); i++) {

char ch = cipherText.charAt(i);

ch = (char) ((ch - 'A' - key + 26) % 26 + 'A');

plainText += ch;

}

System.out.println("Key: " + key + " ==> Plaintext: " + plainText);

}

}

public static int getValidKey(Scanner input) {

int key;

do {

System.out.print("Enter the key (an integer): ");

while (!input.hasNextInt()) {

System.out.println("Invalid input. Please enter an integer.");

input.next(); // Consume the non-integer input

}

key = input.nextInt();

input.nextLine(); // Consume the newline character

} while (key < 0); // You can adjust the validation condition as needed

return key;

}

}

**Input & Output:**

Welcome to the Additive Cipher Program!

Select an option:

1. Encryption

2. Decryption

3. Brute Force

4. Exit

Enter your choice: 1

You selected Encryption. Enter the plaintext: anvesh Enter the key (an integer): 9 The ciphertext is: JWENBO

Select an option:

1. Encryption

2. Decryption

3. Brute Force

4. Exit

Enter your choice: 2

You selected Decryption. Enter the ciphertext: JWENBO

Enter the key (an integer): 9

The plaintext is: ANVESH

Enter your choice: 3

You selected Brute Force.

Enter the ciphertext: JWENBO

Key: 0==> Plaintext: JWENBO

Key: 1==> Plaintext: IVDMAP

Key: 2==> Plaintext: HUCLZO

Key: 3==> Plaintext: GTBKYN

Key: 4 ==> Plaintext: FSAJXM

Key: 5==> Plaintext: ERZIWL

Key: 6 ==> Plaintext: DQYHVK

Key: 7 ==> Plaintext: CPXGUJ

Key: 8> Plaintext: BONFTI

Key: 9 Plaintext: ANVESH

Key: 10==> Plaintext: ZMUDRG

Key: 11==> Plaintext: YLTCOF

Key: 12> Plaintext: XKSBPE

Key: 13> Plaintext: WJRAOD

Key: 14 ==> Plaintext: VIQZNC

Key: 15> Plaintext: UHPYMB

<Key: 16 ==> Plaintext: TGOXLA Key: 17 ==> Plaintext: SFNWKZ

Key: 18> Plaintext: REMVJY

Key: 19> Plaintext: QDLUIX

Key: 20==> Plaintext: PCKTHW

Key: 21 ==> Plaintext: OBJSGV Key: 22 > Plaintext: NAIRFU

Key: 23==> Plaintext: MZHQET

Key: 24> Plaintext: LYGPDS

Key: 25> Plaintext: KXFOCR

Select an option:

1. Encryption

2. Decryption

3. Brute Force

4 Exit

Enter your choice: 4

Goodbye!

**Experiment 2**

Write a program to implement Multiplicative Cipher.

* Plaintext should be in lowercase.
* Ciphertext should be uppercase.
* Brute force attack.

**Source Code:**

import java.util.\*;

class Main {

static void bruteforce()

{

Scanner sc = new Scanner(System.in);

System.out.println("You selected Brute force. ");

System.out.println("Enter the Cipher text. ");

String ciphertext = sc.next();

if(!ciphertext.matches("[A-Z]+"))

{

System.out.println("Cipher text should contains only Upper letter");

return;

}

System.out.print("Enter the expected plaintext: ");

String expectedPlainText = sc.next();

for (int key = 1; key < 26; key++) {

int inverseKey = findinverse(key);

if (inverseKey != -1) {

StringBuilder plaintext = new StringBuilder();

for (int i = 0; i < ciphertext.length(); i++) {

char ch = ciphertext.charAt(i);

if (Character.isLetter(ch)) {

char base = Character.isUpperCase(ch) ? 'A' : 'a';

ch = (char) (base + (ch - base + 26) \* inverseKey % 26);

}

plaintext.append(ch);

}

if (plaintext.toString().equalsIgnoreCase(expectedPlainText)) {

System.out.println("Key: " + key + " ==> Decrypted Text: " + plaintext);

}

}

}

}

static void decryption()

{

Scanner sc = new Scanner(System.in);

System.out.println("You selected Decryption. ");

System.out.println("Enter the Cipher text");

String ciphertext = sc.next();

if(!ciphertext.matches("[A-Z]+"))

{

System.out.println("Cipher text should only contains Upper letters");

return;

}

int key = getValidKey(sc);

int inverseKey = findinverse(key);

if (inverseKey == -1) {

System.out.println("Error: The key has no multiplicative inverse in the alphabet size.");

return;

}

String plaintext = "";

for(int i=0; i<ciphertext.length(); i++)

{

char ch = ciphertext.charAt(i);

ch = (char) ('A' + (ch-'A'+26)\*inverseKey%26);

plaintext+=ch;

}

System.out.println("The plaintext is -"+plaintext.toLowerCase());

}

static void encryption()

{

Scanner sc = new Scanner(System.in);

System.out.println("You selected Encryption -");

System.out.println("Enter the plain text ");

String plaintext = sc.next();

if(!plaintext.matches("[a-z]+"))

{

System.out.println("Plain text should contains only lower Case ");

return;

}

int key = getValidKey(sc);

plaintext = plaintext.toUpperCase();

StringBuilder cipherText = new StringBuilder();

for (int i = 0; i < plaintext.length(); i++) {

char ch = plaintext.charAt(i);

ch = (char) ('A' + (ch - 'A') \* key % 26);

cipherText.append(ch);

}

System.out.println("The ciphertext is: " + cipherText.toString().toUpperCase());

}

public static int getValidKey(Scanner input) {

int key;

do {

System.out.print("Enter the key (an integer): ");

while (!input.hasNextInt()) {

System.out.println("Invalid input. Please enter an integer.");

input.next(); // Consume the non-integer input

}

key = input.nextInt();

input.nextLine(); // Consume the newline character

} while (key < 0); // You can adjust the validation condition as needed

return key;

}

public static int findinverse(int key)

{

for(int i=1; i<26; i++)

{

if((key\*i)%26==1)

{

return i;

}

}

return -1;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Welcome to the Multiplicative Cipher");

int choice = 0;

do

{

System.out.println("Select the option - ");

System.out.println("1. Encytption");

System.out.println("2. Decryption");

System.out.println("3. Brute Force");

System.out.println("4. Exit");

try

{

choice = sc.nextInt();

switch(choice)

{

case 1:

encryption();

break;

case 2:

decryption();

break;

case 3:

bruteforce();

break;

case 4:

System.out.println("Good Bye!!!");

break;

default :

System.out.println("Please enter the valid choice.");

break;

}

}catch(InputMismatchException e)

{

System.out.println("Please Enter the valid choice");

}

}while(choice!=4);

sc.close();

}

}

**Input & Output:**

Welcome to the Multiplicative Cipher Program!

Select an option:

1. Encryption

2. Decryption

3. Brute Force

4. Exit

Enter your choice: 1

You selected Encryption. Enter the plaintext: anvesh

Enter the key (an integer): 9

The ciphertext is: ANHKGL

Select an option:

1. Encryption

2. Decryption

3. Brute Force

4. Exit

Enter your choice: 2

You selected Decryption. Enter the ciphertext: ANHKGL

Enter the key (an integer): 9

The plaintext is: ANVESH

Enter your choice: 3

You selected Brute Force.

Enter the ciphertext: ANHKGL

Key: 9 🡺 Decrypted Text : ANVESH

Select an option:

1. Encryption

2. Decryption

3. Brute Force

4 Exit

Enter your choice: 4

Goodbye

**Experiment 3**

Write a program to implement Affine Cipher.

* Plaintext should be in lowercase.
* Ciphertext should be uppercase.
* Brute force attack.

# Source Code:

import java.util.InputMismatchException;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.println("\nWelcome to the Affine Cipher Program!");

int choice = 0;

do {

System.out.println("\nSelect an option:");

System.out.println("1. Encryption");

System.out.println("2. Decryption");

System.out.println("3. Brute Force Decryption");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

try {

choice = input.nextInt();

input.nextLine();

switch (choice) {

case 1:

encryption();

break;

case 2:

decryption();

break;

case 3:

bruteForceDecryption();

break;

case 4:

System.out.println("\nGoodbye!");

break;

default:

System.out.println("\nInvalid choice. Please choose from 1-4.");

break;

}

} catch (InputMismatchException e) {

System.out.println("\nInvalid input. Please enter a valid choice (1-4).");

input.nextLine(); // Consume the invalid input

}

} while (choice != 4);

input.close();

}

public static void encryption() {

Scanner input = new Scanner(System.in);

System.out.println("\nYou selected Encryption.");

String plainText;

boolean validInput;

do {

System.out.print("Enter the plaintext (lowercase alphabetic characters only): ");

plainText = input.nextLine();

validInput = plainText.matches("[a-z ]+");

if (!validInput) {

System.out.println("Error: The plaintext should only contain lowercase alphabetic characters.");

}

} while (!validInput);

int keyA, keyB;

do {

System.out.print("Enter key A (an integer coprime to 26): ");

while (!input.hasNextInt()) {

System.out.println("Invalid input. Please enter an integer.");

input.next();

}

keyA = input.nextInt();

input.nextLine();

} while (!isCoprime(keyA, 26));

System.out.print("Enter key B (an integer): ");

while (!input.hasNextInt()) {

System.out.println("Invalid input. Please enter an integer.");

input.next();

}

keyB = input.nextInt();

input.nextLine();

StringBuilder cipherText = new StringBuilder();

for (int i = 0; i < plainText.length(); i++) {

char ch = plainText.charAt(i);

if (Character.isLetter(ch)) {

char base = 'a';

ch = (char) (((ch - base) \* keyA + keyB) % 26 + base);

}

cipherText.append(ch);

}

System.out.println("The ciphertext is: " + cipherText.toString().toUpperCase());

// Close the scanner when done

return;

}

public static void decryption() {

Scanner input = new Scanner(System.in);

System.out.println("\nYou selected Decryption.");

String cipherText;

boolean validInput;

do {

System.out.print("Enter the ciphertext (uppercase alphabetic characters only): ");

cipherText = input.nextLine();

validInput = cipherText.matches("[A-Z ]+"); // Check if input is valid

if (!validInput) {

System.out.println("Error: The ciphertext should only contain uppercase alphabetic characters.");

}

} while (!validInput);

int keyA, keyB;

do {

System.out.print("Enter key A (an integer coprime to 26): ");

while (!input.hasNextInt()) {

System.out.println("Invalid input. Please enter an integer.");

input.next();

}

keyA = input.nextInt();

input.nextLine();

} while (!isCoprime(keyA, 26));

System.out.print("Enter key B (an integer): ");

while (!input.hasNextInt()) {

System.out.println("Invalid input. Please enter an integer.");

input.next();

}

keyB = input.nextInt();

input.nextLine();

int inverseKeyA = findInverse(keyA, 26);

if (inverseKeyA == -1) {

System.out.println("Error: Key A has no multiplicative inverse in the alphabet size.");

return;

}

StringBuilder plainText = new StringBuilder();

for (int i = 0; i < cipherText.length(); i++) {

char ch = cipherText.charAt(i);

if (Character.isLetter(ch)) {

char base = 'A';

ch = (char) (((ch - base - keyB + 26) \* inverseKeyA) % 26 + base);

}

plainText.append(ch);

}

System.out.println("The plaintext is: " + plainText);

// Close the scanner when done

return;

}

public static void bruteForceDecryption() {

Scanner input = new Scanner(System.in);

System.out.println("\nYou selected Brute Force Decryption.");

String cipherText;

boolean validInput;

do {

System.out.print("Enter the ciphertext (uppercase alphabetic characters only): ");

cipherText = input.nextLine();

validInput = cipherText.matches("[A-Z ]+"); // Check if input is valid

if (!validInput) {

System.out.println("Error: The ciphertext should only contain uppercase alphabetic characters.");

}

} while (!validInput);

System.out.print("Enter the known plaintext: ");

String knownPlainText = input.nextLine();

System.out.println("Attempting brute force decryption...");

for (int keyA = 1; keyA < 26; keyA++) {

for (int keyB = 0; keyB < 26; keyB++) {

int inverseKeyA = findInverse(keyA, 26);

StringBuilder plainText = new StringBuilder();

for (int i = 0; i < cipherText.length(); i++) {

char ch = cipherText.charAt(i);

if (Character.isLetter(ch)) {

char base = 'A';

ch = (char) (((ch - base - keyB + 26) \* inverseKeyA) % 26 + base);

}

plainText.append(ch);

}

if (plainText.toString().equals(knownPlainText)) {

System.out.println("Key A: " + keyA + ", Key B: " + keyB + " - " + knownPlainText);

}

}

}

return;

}

public static boolean isCoprime(int a, int b) {

// Check if two numbers are coprime (have gcd = 1)

while (b != 0) {

int temp = b;

b = a % b;

a = temp;

}

return a == 1;

}

public static int findInverse(int a, int m) {

for (int x = 1; x < m; x++) {

if ((a \* x) % m == 1) {

return x;

}

}

return -1; // No modular inverse exists

}

}

**Input & Output:**

Welcome to the Affine Cipher Program!

Select an option:

1. Encryption

2. Decryption

3. Brute Force Decryption

4. Exit

Enter your choice: 1

You selected Encryption.

Enter the plaintext (lowercase alphabetic characters only): anvesh

Enter key A (an integer coprime to 26): 9

Enter key B (an integer): 8

The ciphertext is: IVPSOT

Select an option:

1. Encryption

2. Decryption

3. Brute Force Decryption

4. Exit

Enter your choice: 2

You selected Decryption.

Enter the ciphertext (uppercase alphabetic characters only): IVPSOT

Enter key A (an integer coprime to 26): 9

Enter key B (an integer): 8

The plaintext is: ANVESH

Select an option:

1. Encryption

2. Decryption

3. Brute Force Decryption

4. Exit

Enter your choice: 3

You selected Brute Force Decryption.

Enter the ciphertext (uppercase alphabetic characters only): IVPSOT

Enter the known plaintext: anvesh

Attempting brute force decryption...

Select an option:

1. Encryption

2. Decryption

3. Brute Force Decryption

4. Exit

Enter your choice: 4

Goodbye!

**Experiment 4**

Write a program in to implement Autokey Cipher.

* Plaintext should be in lowercase.
* Ciphertext should be uppercase.
* Brute force attack.

# Source Code:

import java.util.InputMismatchException;

import java.util.Scanner;

public class Main {

public static void bruteForce() {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the cipher text");

String ciphertext = sc.nextLine();

if (!ciphertext.matches("^[A-Z ]+$")) {

System.out.println("Error: The ciphertext should only contain upper letters.");

return;

}

System.out.print("Enter the expected plaintext: ");

String expectedPlainText = sc.next();

for (int key = 1; key < 26; key++) {

String ans = "";

int newkey = key;

for (int i = 0; i < ciphertext.length(); i++) {

char ch = ciphertext.charAt(i);

if(ch==' ') {

ans+=' ';

continue;

}

ch = (char)(ch-'A');

ch = (char) ('A'+(ch-newkey)%26);

if((ch-65)<0)

{

ch+=26;

}

ans+=ch;

newkey = ch-65;

}

if (ans.equalsIgnoreCase(expectedPlainText)) {

System.out.println("Key: " + key + " ==> Decrypted Text: " + ans);

}

}

}

public static void decryption() {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the cipher text");

String ciphertext = sc.nextLine();

if (!ciphertext.matches("[A-Z ]+")) {

System.out.println("Error: The ciphertext should only contain upper letters.");

return;

}

int key = 0;

boolean isValidInput = false;

while (!isValidInput) {

try {

System.out.print("Enter an key: ");

key = sc.nextInt();

isValidInput = true;

} catch (java.util.InputMismatchException e) {

System.out.println("Invalid input. Please enter a valid integer.");

sc.nextLine();

}

}

String ans = "";

for (int i = 0; i < ciphertext.length(); i++) {

char ch = ciphertext.charAt(i);

if(ch==' ') {

ans+=' ';

continue;

}

ch = (char)(ch-'A');

ch = (char) ('A'+(ch-key)%26);

if((ch-65)<0)

{

ch+=26;

}

ans+=ch;

key = ch-65;

}

System.out.println(ans.toLowerCase());

}

public static void encryption()

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter the plain text");

String plaintext = sc.nextLine();

if (!plaintext.matches("[a-z ]+")) {

System.out.println("Error: The plaintext should only contain lower letters.");

return;

}

int key = 0;

boolean isValidInput = false;

while (!isValidInput) {

try {

System.out.print("Enter an key: ");

key = sc.nextInt();

isValidInput = true;

} catch (java.util.InputMismatchException e) {

System.out.println("Invalid input. Please enter a valid integer.");

sc.nextLine();

}

}

String ans = "";

for (int i = 0; i <plaintext.length(); i++) {

int ch = plaintext.codePointAt(i);

if(ch==' ')

{

ans+=' ';

continue;

}

int newkey = plaintext.codePointAt(i);

ch=ch-97;

ch=((ch+key)%26);

char newChar = (char) (ch+97 );

ans+=newChar;

key=newkey-97;

}

System.out.println(ans.toUpperCase());

}

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int choice = 0;

do {

System.out.println("\nSelect an option:");

System.out.println("1. Encryption");

System.out.println("2. Decryption");

System.out.println("3. Brute Force");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

try {

choice = input.nextInt();

input.nextLine(); // Consume the newline character

switch (choice) {

case 1:

encryption();

break;

case 2:

decryption();

break;

case 3:

bruteForce();

break;

case 4:

System.out.println("\nGoodbye!");

break;

default:

System.out.println("\nInvalid choice. Please choose from 1-4.");

break;

}

} catch (InputMismatchException e) {

System.out.println("\nInvalid input. Please enter a valid choice (1-4).");

input.nextLine(); // Consume the invalid input

}

} while (choice != 4);

input.close();

}

public static int getValidKey(Scanner input) {

int key;

do {

System.out.print("Enter the key (an integer): ");

while (!input.hasNextInt()) {

System.out.println("Invalid input. Please enter an integer.");

input.next(); // Consume the non-integer input

}

key = input.nextInt();

input.nextLine(); // Consume the newline character

} while (key < 0); // You can adjust the validation condition as needed

return key;

}

}

**Input & Output:**

Welcome to the Auto key Cipher Program!

Select an option:

1. Encryption

2. Decryption

3. Brute Force Decryption

4. Exit

Enter your choice: 1

You selected Encryption.

Enter the plaintext (lowercase alphabetic characters only): anvesh

Enter key A (an integer coprime to 26): 9

Enter key B (an integer): 8

The ciphertext is: IVPSOT

Select an option:

1. Encryption

2. Decryption

3. Brute Force Decryption

4. Exit

Enter your choice: 2

You selected Decryption.

Enter the ciphertext (uppercase alphabetic characters only): IVPSOT

Enter key A (an integer coprime to 26): 9

Enter key B (an integer): 8

The plaintext is: ANVESH

Select an option:

1. Encryption

2. Decryption

3. Brute Force Decryption

4. Exit

Enter your choice: 3

You selected Brute Force Decryption.

Enter the ciphertext (uppercase alphabetic characters only): IVPSOT

Enter the known plaintext: anvesh

Attempting brute force decryption...

Select an option:

1. Encryption

2. Decryption

3. Brute Force Decryption

4. Exit

Enter your choice: 4

Goodbye!

**Experiment 5**

Write a program to implement Playfair Cipher to encrypt & decrypt the given message where the key matrix can be formed by using a given keyword.

# Source Code:

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.println("\nWelcome to the Playfair Cipher Program!");

int choice = 0;

do {

System.out.println("\nSelect an option:");

System.out.println("1. Encryption");

System.out.println("2. Decryption");

System.out.println("3. Exit");

System.out.print("Enter your choice: ");

choice = input.nextInt();

input.nextLine();

switch (choice) {

case 1:

encryption();

break;

case 2:

decryption();

break;

case 3:

System.out.println("\nGoodbye!");

break;

default:

System.out.println("\nInvalid choice. Please choose from 1-3.");

break;

}

} while (choice != 3);

input.close();

}

public static void encryption() {

Scanner input = new Scanner(System.in);

System.out.println("\nYou selected Encryption.");

System.out.print("Enter the encryption key: ");

String key = input.nextLine().toUpperCase(); // Convert to uppercase

char[][] keyMatrix = createKeyMatrix(key);

// Display the key matrix

System.out.println("Key Matrix:");

displayMatrix(keyMatrix);

System.out.print("Enter the plaintext (letters only): ");

String plainText = input.nextLine().toUpperCase().replaceAll("[^A-Z]", ""); // Remove non-uppercase letters

String cipherText = playfairEncrypt(plainText, keyMatrix);

System.out.println("The ciphertext is: " + cipherText);

}

public static void decryption() {

Scanner input = new Scanner(System.in);

System.out.println("\nYou selected Decryption.");

System.out.print("Enter the decryption key: ");

String key = input.nextLine().toUpperCase(); // Convert to uppercase

char[][] keyMatrix = createKeyMatrix(key);

// Display the key matrix

System.out.println("Key Matrix:");

displayMatrix(keyMatrix);

System.out.print("Enter the ciphertext (letters only): ");

String cipherText = input.nextLine().toUpperCase().replaceAll("[^A-Z]", ""); // Remove non-uppercase letters

String plainText = playfairDecrypt(cipherText, keyMatrix);

System.out.println("The plaintext is: " + plainText);

}

public static void displayMatrix(char[][] matrix) {

for (int i = 0; i < 5; i++) {

for (int j = 0; j < 5; j++) {

System.out.print(matrix[i][j] + " ");

}

System.out.println();

}

}

public static char[][] createKeyMatrix(String key) {

char[][] matrix = new char[5][5];

String keyWithoutDuplicates = removeDuplicates(key + "ABCDEFGHIKLMNOPQRSTUVWXYZ");

int row = 0, col = 0;

for (int i = 0; i < keyWithoutDuplicates.length(); i++) {

char ch = keyWithoutDuplicates.charAt(i);

matrix[row][col] = ch;

col++;

if (col == 5) {

row++;

col = 0;

}

}

return matrix;

}

public static String removeDuplicates(String str) {

StringBuilder result = new StringBuilder();

for (int i = 0; i < str.length(); i++) {

char ch = str.charAt(i);

if (result.indexOf(String.valueOf(ch)) == -1) {

result.append(ch);

}

}

return result.toString();

}

public static String playfairEncrypt(String plainText, char[][] keyMatrix) {

StringBuilder cipherText = new StringBuilder();

for (int i = 0; i < plainText.length(); i += 2) {

char firstChar = plainText.charAt(i);

char secondChar = (i + 1 < plainText.length()) ? plainText.charAt(i + 1) : 'X';

int[] firstCharPos = findCharPosition(firstChar, keyMatrix);

int[] secondCharPos = findCharPosition(secondChar, keyMatrix);

int row1 = firstCharPos[0];

int col1 = firstCharPos[1];

int row2 = secondCharPos[0];

int col2 = secondCharPos[1];

if (row1 == row2) {

col1 = (col1 + 1) % 5;

col2 = (col2 + 1) % 5;

} else if (col1 == col2) {

row1 = (row1 + 1) % 5;

row2 = (row2 + 1) % 5;

} else {

int temp = col1;

col1 = col2;

col2 = temp;

}

cipherText.append(keyMatrix[row1][col1]);

cipherText.append(keyMatrix[row2][col2]);

}

return cipherText.toString();

}

public static String playfairDecrypt(String cipherText, char[][] keyMatrix) {

StringBuilder plainText = new StringBuilder();

for (int i = 0; i < cipherText.length(); i += 2) {

char firstChar = cipherText.charAt(i);

char secondChar = cipherText.charAt(i + 1);

int[] firstCharPos = findCharPosition(firstChar, keyMatrix);

int[] secondCharPos = findCharPosition(secondChar, keyMatrix);

int row1 = firstCharPos[0];

int col1 = firstCharPos[1];

int row2 = secondCharPos[0];

int col2 = secondCharPos[1];

if (row1 == row2) {

col1 = (col1 - 1 + 5) % 5;

col2 = (col2 - 1 + 5) % 5;

} else if (col1 == col2) {

row1 = (row1 - 1 + 5) % 5;

row2 = (row2 - 1 + 5) % 5;

} else {

int temp = col1;

col1 = col2;

col2 = temp;

}

plainText.append(keyMatrix[row1][col1]);

plainText.append(keyMatrix[row2][col2]);

}

return plainText.toString();

}

public static int[] findCharPosition(char ch, char[][] keyMatrix) {

int[] position = new int[2];

for (int i = 0; i < 5; i++) {

for (int j = 0; j < 5; j++) {

if (keyMatrix[i][j] == ch) {

position[0] = i;

position[1] = j;

return position;

}

}

}

return position;

}

}

**Input & Output:**

Welcome to the Playfair Cipher Program!

Select an option:

1. Encryption

2. Decryption

3. Brute Force Decryption

4. Exit

Enter your choice: 1

You selected Encryption.

Enter the plaintext (lowercase alphabetic characters only): anvesh

Enter key A (an integer coprime to 26): 9

Enter key B (an integer): 8

The ciphertext is: IVPSOT

Select an option:

1. Encryption

2. Decryption

3. Brute Force Decryption

4. Exit

Enter your choice: 2

You selected Decryption.

Enter the ciphertext (uppercase alphabetic characters only): IVPSOT

Enter key A (an integer coprime to 26): 9

Enter key B (an integer): 8

The plaintext is: ANVESH

Select an option:

1. Encryption

2. Decryption

3. Brute Force Decryption

4. Exit

Enter your choice: 3

You selected Brute Force Decryption.

Enter the ciphertext (uppercase alphabetic characters only): IVPSOT

Enter the known plaintext: anvesh

Attempting brute force decryption...

Select an option:

1. Encryption

2. Decryption

3. Brute Force Decryption

4. Exit

Enter your choice: 4

Goodbye!

**Experiment 6**

Write a program to implement Hill Cipher to encrypt & decrypt the given message by using a given key matrix. Show the values for key and its corresponding key inverse values.

**Source Code:**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Hill Cipher Implementation");

System.out.println("Enter the key matrix size (n xn): ");

int n = scanner.nextInt();

int[][] keyMatrix = new int[n][n];

System.out.println("Enter the key matrix elements:");

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

for (int j = 0; j < n; j++) {

keyMatrix[i][j] = scanner.nextInt() % 26;

}

}

int[][] keyInverse = getKeyInverse(keyMatrix, n);

System.out.println("Key Matrix:");

displayMatrix(keyMatrix);

System.out.println("Key Inverse:");

displayMatrix(keyInverse);

System.out.println("Enter the plaintext (in uppercase):");

scanner.nextLine();

String input = scanner.nextLine();

String ciphertext = encrypt(input, keyMatrix, n);

System.out.println("Encrypted Text: " + ciphertext);

String decryptedText = decrypt(ciphertext, keyInverse, n);

System.out.println("Decrypted Text:" + decryptedText);

}

private static String encrypt(String plaintext, int[][] keyMatrix, int n) {

StringBuilder ciphertext = new StringBuilder();

while (plaintext.length() % n != 0) {

plaintext += 'X';

}

for (int i = 0; i < plaintext.length(); i += n) {

String block = plaintext.substring(i, i + n);

for (int j = 0; j < n; j++) {

int sum = 0;

for (int k = 0; k < n; k++) {

sum += (keyMatrix[j][k] \* (block.charAt(k) - 'A')) % 26;

sum = sum % 26;

}

ciphertext.append((char) ('A' + sum));

}

}

return ciphertext.toString();

}

private static String decrypt(String ciphertext, int[][] keyInverse, int n) {

StringBuilder plaintext = new StringBuilder();

for (int i = 0; i < ciphertext.length(); i += n) {

String block = ciphertext.substring(i, i + n);

for (int j = 0; j < n; j++) {

int sum = 0;

for (int k = 0; k < n; k++) {

sum += (keyInverse[j][k] \* (block.charAt(k) - 'A' + 26)) % 26;

sum%=26;

}

plaintext.append((char) ('A' + sum));

}

}

return plaintext.toString();

}

private static int[][] getKeyInverse(int[][] keyMatrix, int n) {

int det = determinant(keyMatrix, n);

int detInverse = modInverse(det, 26);

int[][] adjugate = adjugate(keyMatrix, n);

int[][] keyInverse = new int[n][n];

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

for (int j = 0; j < n; j++) {

keyInverse[i][j] = (adjugate[i][j] \* detInverse) % 26;

if (keyInverse[i][j] < 0) {

keyInverse[i][j] += 26;

}

}

}

return keyInverse;

}

private static int determinant(int[][] matrix, int n) {

if (n == 1) {

return matrix[0][0];

}

if (n == 2) {

return (matrix[0][0] \* matrix[1][1] - matrix[0][1] \* matrix[1][0] + 26) % 26;

}

int det = 0;

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

int[][] submatrix = new int[n - 1][n - 1];

for (int j = 1; j < n; j++) {

for (int k = 0, l = 0; k < n; k++) {

if (k != i) {

submatrix[j - 1][l++] = matrix[j][k];

}

}

}

int sign = (i % 2 == 0) ? 1 : -1;

det = (det + sign \* matrix[0][i] \* determinant(submatrix, n - 1) + 26) % 26;

}

return det;

}

private static int[][] adjugate(int[][] matrix, int n) {

int[][] adjugate = new int[n][n];

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

for (int j = 0; j < n; j++) {

int[][] submatrix = new int[n - 1][n - 1];

for (int k = 0, l = 0; k < n; k++) {

if (k != i) {

for (int m = 0, n1 = 0; m < n; m++) {

if (m != j) {

submatrix[l][n1++] = matrix[k][m];

}

}

l++;

}

}

int sign = ((i + j) % 2 == 0) ? 1 : -1;

adjugate[j][i] = (sign \* determinant(submatrix, n - 1) + 26) % 26;

}

}

return adjugate;

}

private static int modInverse(int a, int m) {

a = a % m;

for (int x = 1; x < m; x++) {

if ((a \* x) % m == 1) {

return x;

}

}

return 1;

}

private static void displayMatrix(int[][] matrix) {

for (int[] row : matrix) {

for (int element : row) {

System.out.print(element + " ");

}

System.out.println();

}

System.out.println();

}

}

**Input & Output:**

Hill Cipher Implementation Enter the key matrix size (n xn):

3 3

Enter the key matrix elements:

1 2 3 4 5 6 7 8 9

Key Matrix:

3 1 2

3 4 5

6 7 8

Key Inverse:

9 8 9

8 16 1

9 19 25

Enter the plaintext (in uppercase):

ANVESHMANGALAM

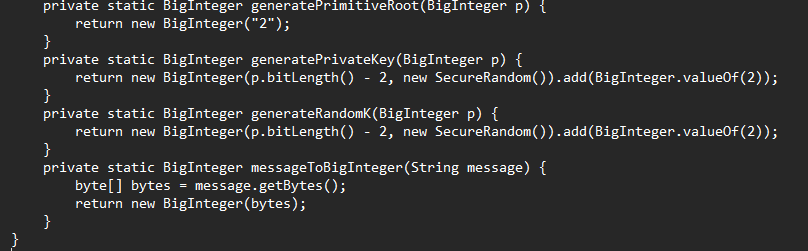
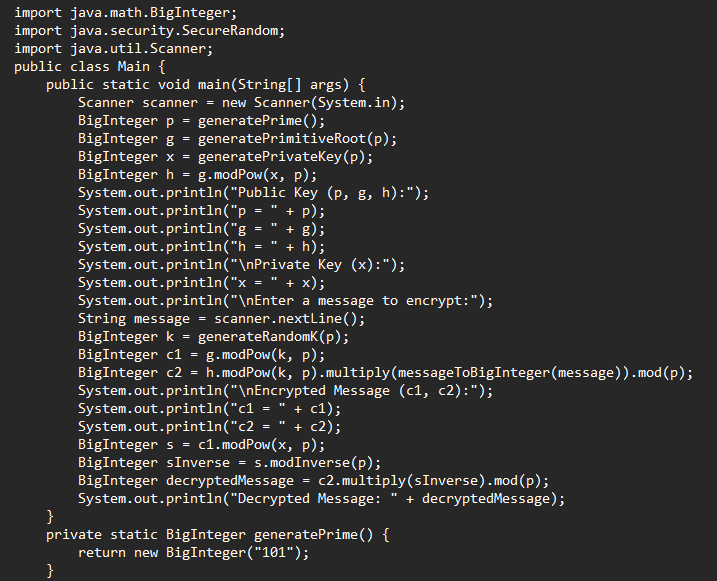
Encrypted Text: DBZSPYKXUOVUGHI

US. GDB Decrypted Text:ANVESHMANGALAMX

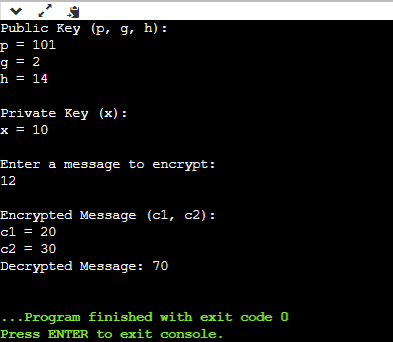
**Experiment 7**

Write a program to implement Elgamal Cryptosystem to generate the pair of keys and then show the encryption & decryption of a given message.

**Source Code:**



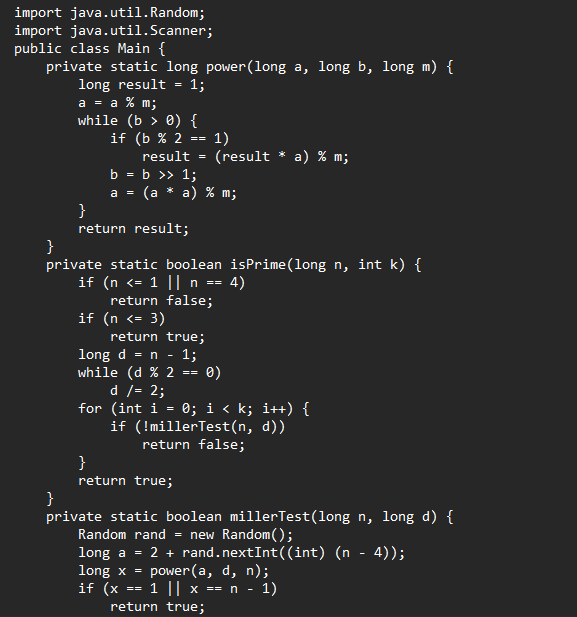
**Input & Ouput:**

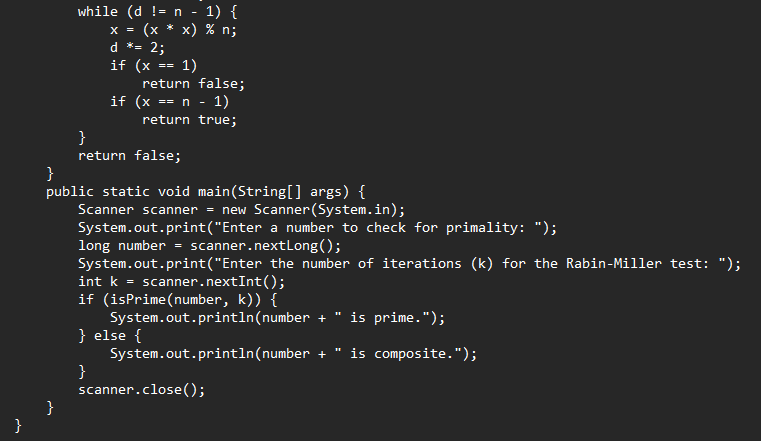


**Experiment 8**

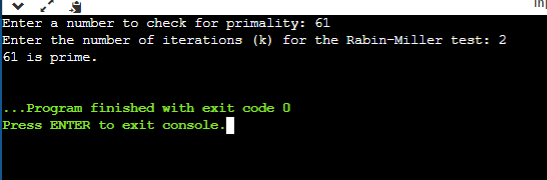
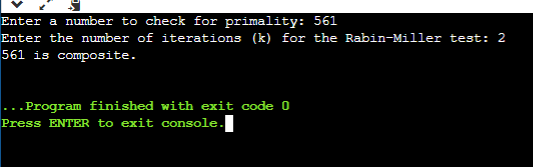
Write a program to implement Rabin Miller Primality Test to check whether given number is prime or composite.

**Source Code:**





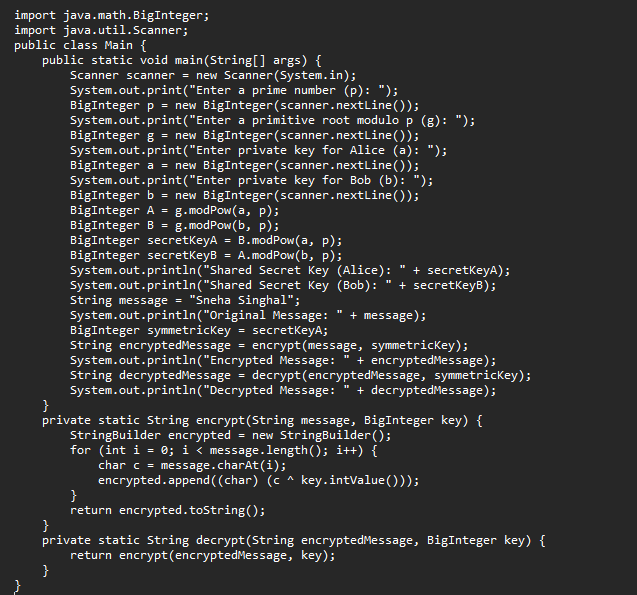
**Input & Output:**



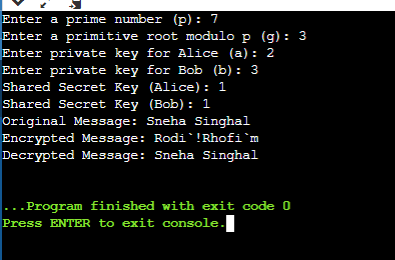
**Experiment 9**

Write a program to implement Diffie-Hellman key exchange Algorithm to exchange the symmetric key and show the encryption & decryption.

# Source Code:



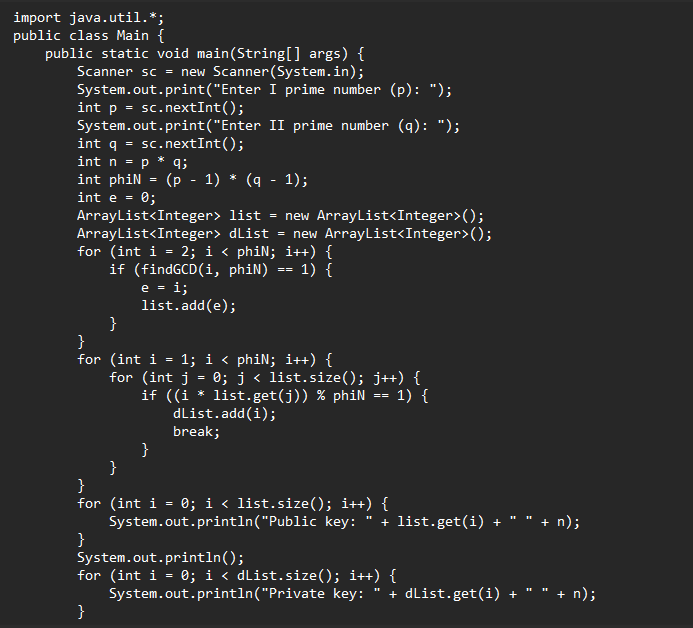
**Input & Output:**

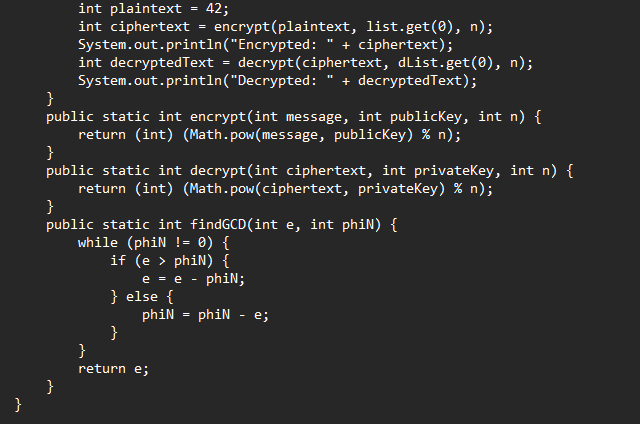


**Experiment 10**

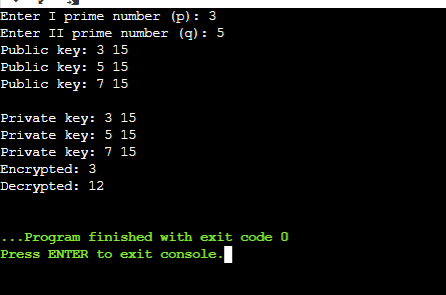
Write a program to implement RSA Algorithm to generate a pair of keys and show the encryption and decryption by using a given key pair.

# Source Code:





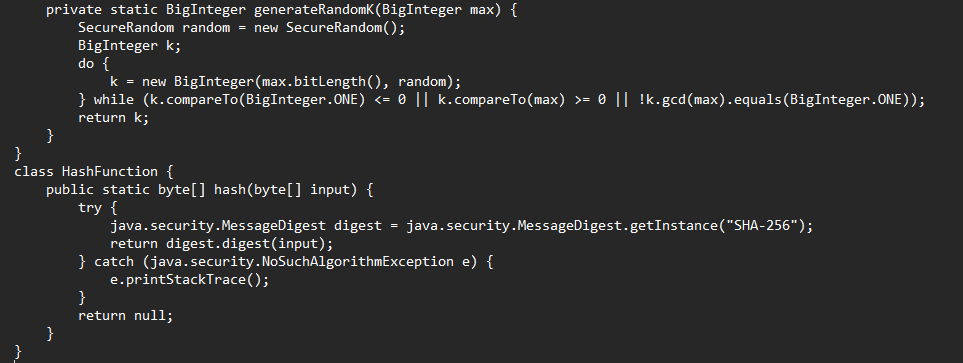
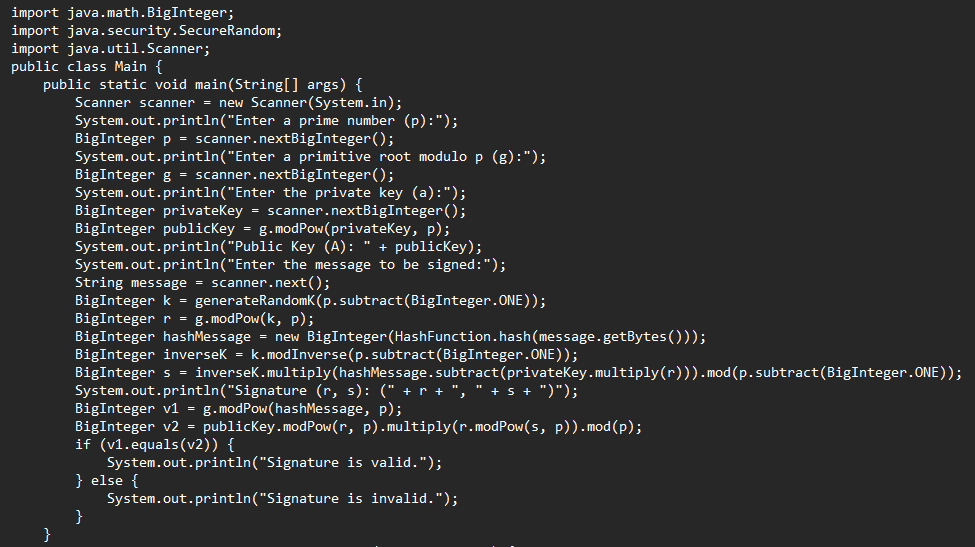
**Input & Output:**



**Experiment 11**

Write a program to implement Elgamal algorithm for implementing digital signature.

**Source Code:**



**Input & Output:**

