1.

**Aim:**

To compute adjoint 3x3 matrix in modular arithematic and compute co factor of each element.

**Procedure:**

1. Get 3x3 matrix from user.

2. Calculator adjoint for the matrix and co factor for each element.

3. Display the results in console.

**Code:**

package com.yajith.adjoint;

import java.util.Scanner;

public class Main {

static final int N=3;

static void getCofactor(int A[][],int temp[][],int p,int q,int n)

{

int i=0,j=0;

for(int row=0;row<n;row++)

{

for(int col=0;col<n;col++)

{

if(row!=p&&col!=q)

{

temp[i][j++]=A[row][col];

if(j==n-1)

{

j=0;i++;

}

}

}

}

}

static int determinant(int A[][],int n)

{

int D=0;

if(n==1)

{

return A[0][0];

}

int [][] temp=new int[N][N];

int sign=1;

for(int f=0;f<n;f++)

{

getCofactor(A,temp,0,f,n);

D+=sign\*A[0][f]\*determinant(temp,n-1);

sign=-sign;

}

return D;

}

static void adjoint(int A[][] ,int adj[][])

{

if(N==1)

{

adj[0][0]=1;

return;

}

int sign=1;

int temp[][]=new int[N][N];

for(int i=0;i<N;i++)

{

for(int j=0;j<N;j++)

{

getCofactor(A,temp,i,j,N);

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

adj[j][i]=(sign)\*determinant(temp,N-1);

}

}

}

public static void main(String[] args) {

int[][] A=new int[N][N];

int [][] adj=new int[N][N];

Scanner scanner=new Scanner(System.in);

System.out.println("Enter elements of matrix");

for(int i=0; i<N; i++)

{

for(int j=0; j<N; j++)

{

A[i][j]=scanner.nextInt();

}

}

adjoint(A,adj);

System.out.println("Cofactors are");

{

for(int i=0; i<N; i++)

{

for(int j=0; j<N; j++)

{

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

}

System.out.println();

}

}

System.out.println("Adjoints are");

for(int i=0; i<N; i++)

{

for(int j=0; j<N; j++)

{

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

}

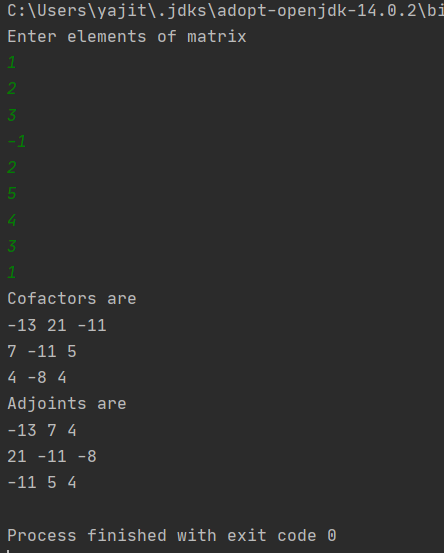
System.out.println();

}

}

}

**Output:**



**Result:**

The adjoint and cofactor of 3x3 matrix is verified.

2.

**Aim:**

To create a file and do encryption and decryption using openssl.

**Procedure:**

1. Store "EVOLVE INTO A CENTRE OF EXCELLENCE FOR EDUCATION AND RESEARCH IN INFORMATION TECHNOLOGY" in plain text file.

2. Generate private and public key for RSA.

3. Display the private key in hexadecimal.

4. Perform encryption using RSA public key.

5. Display the contents of encrypted file.

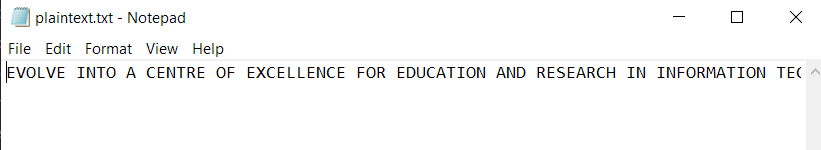
6. Decrypt the result using RSA private key.

7. Display content of decrypted file.

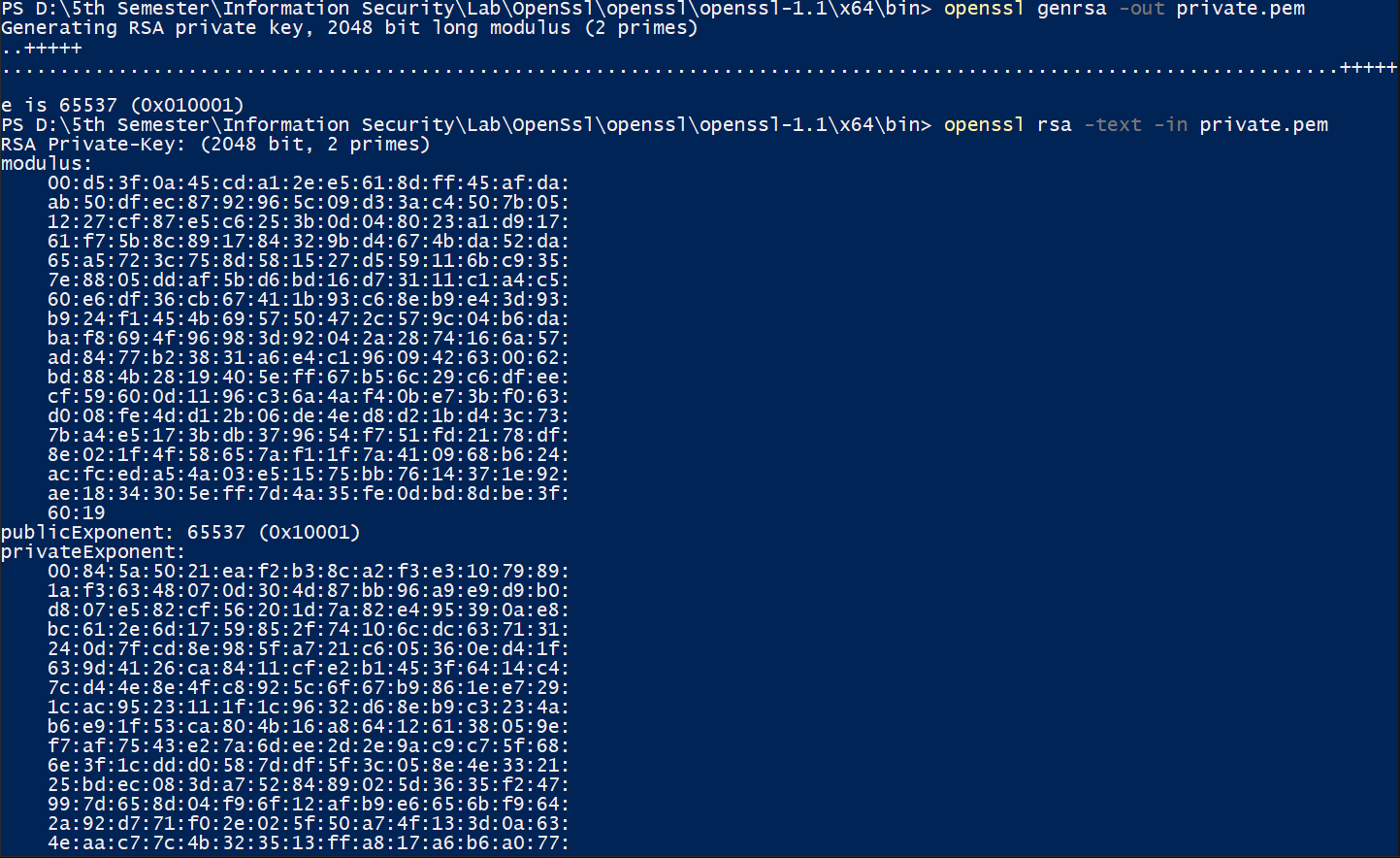
8. Check whether the plain text is same or not.

**Output**:

Content of plain text

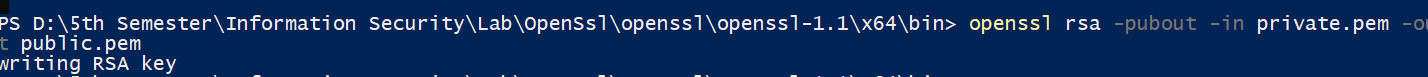


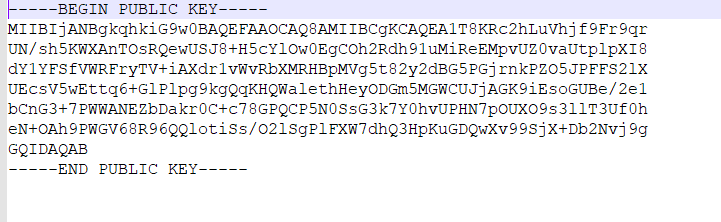
openssl genrsa –out private.pem

openssl rsa -text -in private.pem

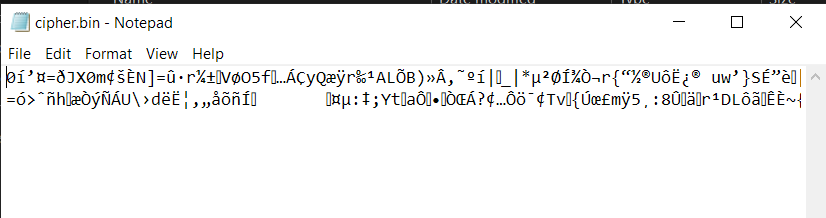


openssl rsa -pubout -in private.pem -out public.pem

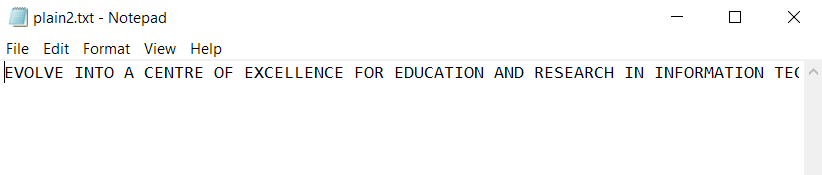




openssl rsautl -encrypt -in plaintext.txt -pubin -inkey public.pem -out cipher.bin



openssl rsautl -decrypt -in c1.bin -inkey private.pem -out plain2.txt



**Result:**

The encryption and decryption using RSA is verified using openssl.