**1(a). Caesar Cipher**

Abdul Rahman S - 7881

class caesarCipher {

public static String encode(String enc, int offset) {

offset = offset % 26 + 26;

StringBuilder encoded = new StringBuilder();

for (char i : enc.toCharArray()) {

if (Character.isLetter(i)) {

if (Character.isUpperCase(i)) {

encoded.append((char) ('A' + (i - 'A' + offset) % 26));

} else {

encoded.append((char) ('a' + (i - 'a' + offset) % 26));

}

} else {

encoded.append(i);

}

}

return encoded.toString();

}

public static String decode(String enc, int offset) {

return encode(enc, 26 - offset);

}

public static void main(String[] args) throws java.lang.Exception {

String msg = "Transport";

System.out.println("Simulating Caesar Cipher\n------------------------");

System.out.println("Input : " + msg);

System.out.printf("Encrypted Message : ");

System.out.println(caesarCipher.encode(msg, 3));

System.out.printf("Decrypted Message : ");

System.out.println(caesarCipher.decode(caesarCipher.encode(msg, 3), 3)); }}

OUTPUT:



**1(b).Hill Cipher**

Abdul Rahman S - 7881

class hillCipher {

/\* 3x3 key matrix for 3 characters at once \*/

public static int[][] keymat = new int[][] { { 1, 2, 1 }, { 2, 3, 2 },

{ 2, 2, 1 } }; /\* key inverse matrix \*/

public static int[][] invkeymat = new int[][] { { -1, 0, 1 }, { 2, -1, 0 }, { -2, 2, -1

} };

public static String key = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

private static String encode(char a, char b, char c) {

String ret = "";

int x, y, z;

int posa = (int) a - 65;

int posb = (int) b - 65;

int posc = (int) c - 65;

x = posa \* keymat[0][0] + posb \* keymat[1][0] + posc \* keymat[2][0];

y = posa \* keymat[0][1] + posb \* keymat[1][1] + posc \* keymat[2][1];

z = posa \* keymat[0][2] + posb \* keymat[1][2] + posc \* keymat[2][2];

a = key.charAt(x % 26);

b = key.charAt(y % 26);

c = key.charAt(z % 26);

ret = "" + a + b + c;

return ret;

}

private static String decode(char a, char b, char c) {

String ret = "";

int x, y, z;

int posa = (int) a - 65;

int posb = (int) b - 65;

int posc = (int) c - 65;

x = posa \* invkeymat[0][0] + posb \* invkeymat[1][0] + posc \*

invkeymat[2][0];

y = posa \* invkeymat[0][1] + posb \* invkeymat[1][1] + posc \*

invkeymat[2][1];

z = posa \* invkeymat[0][2] + posb \* invkeymat[1][2] + posc \*

invkeymat[2][2];

a = key.charAt((x % 26 < 0) ? (26 + x % 26) : (x % 26));

b = key.charAt((y % 26 < 0) ? (26 + y % 26) : (y % 26));

c = key.charAt((z % 26 < 0) ? (26 + z % 26) : (z % 26));

ret = "" + a + b + c;

return ret;

}

public static void main(String[] args) throws java.lang.Exception {

String msg;

String enc = "";

String dec = "";

int n;

msg = ("Information");

System.out.println("simulation of Hill Cipher\n-------------------------");

System.out.println("Input message : " + msg);

msg = msg.toUpperCase();

msg = msg.replaceAll("\\s", "");

/\* remove spaces \*/ n = msg.length() % 3;

/\* append padding text X \*/ if (n != 0) {

for (int i = 1; i <= (3 - n); i++) {

msg += 'X';

}

}

System.out.println("padded message : " + msg);

char[] pdchars = msg.toCharArray();

for (int i = 0; i < msg.length(); i += 3) {

enc += encode(pdchars[i], pdchars[i + 1], pdchars[i + 2]);

}

System.out.println("encoded message : " + enc);

char[] dechars = enc.toCharArray();

for (int i = 0; i < enc.length(); i += 3) {

dec += decode(dechars[i], dechars[i + 1], dechars[i + 2]);

}

System.out.println("decoded message : " + dec);

}

}

OUTPUT:



**1(c). Playfair Cipher**

Abdul Rahman S - 7881

import java.awt.Point;

class playfairCipher {

private static char[][] charTable;

private static Point[] positions;

private static String prepareText(String s, boolean chgJtoI) {

s = s.toUpperCase().replaceAll("[^A-Z]", "");

return chgJtoI ? s.replace("J", "I") : s.replace("Q", "");

}

private static void createTbl(String key, boolean chgJtoI) {

charTable = new char[5][5];

positions = new Point[26];

String s = prepareText(key + "ABCDEFGHIJKLMNOPQRSTUVWXYZ",

chgJtoI);

int len = s.length();

for (int i = 0, k = 0; i < len; i++) {

char c = s.charAt(i);

if (positions[c - 'A'] == null) {

charTable[k / 5][k % 5] = c;

positions[c - 'A'] = new Point(k % 5, k / 5);

k++;

}

}

}

private static String codec(StringBuilder txt, int dir) {

int len = txt.length();

for (int i = 0; i < len; i += 2) {

char a = txt.charAt(i);

char b = txt.charAt(i + 1);

int row1 = positions[a - 'A'].y;

int row2 = positions[b - 'A'].y;

int col1 = positions[a - 'A'].x;

int col2 = positions[b - 'A'].x;

if (row1 == row2) {

col1 = (col1 + dir) % 5;

col2 = (col2 + dir) % 5;

} else if (col1 == col2) {

row1 = (row1 + dir) % 5;

row2 = (row2 + dir) % 5;

} else {

int tmp = col1;

col1 = col2;

col2 = tmp;

}

txt.setCharAt(i, charTable[row1][col1]);

txt.setCharAt(i + 1, charTable[row2][col2]);

}

return txt.toString();

}

private static String encode(String s) {

StringBuilder sb = new StringBuilder(s);

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

if (i == sb.length() - 1) {

sb.append(sb.length() % 2 == 1 ? 'X' : "");

} else if (sb.charAt(i) == sb.charAt(i + 1)) {

sb.insert(i + 1, 'X');

}

}

return codec(sb, 1);

}

private static String decode(String s) {

return codec(new StringBuilder(s), 4);

}

public static void main(String[] args) throws java.lang.Exception {

String key = "CSE";

String txt = "Communication Lab"; /\* make sure string length is even \*/ /\* change J

to I \*/

boolean chgJtoI = true;

createTbl(key, chgJtoI);

String enc = encode(prepareText(txt, chgJtoI));

System.out.println("Simulating Playfair Cipher\n----------------------");

System.out.println("Input Message : " + txt);

System.out.println("Encrypted Message : " + enc);

System.out.println("Decrypted Message : " + decode(enc));

}

}



**1d. Vigenere Cipher**

Abdul Rahman S - 7881

public class vigenereCipher {

static String encode(String text, final String key) {

String res = "";

text = text.toUpperCase();

for (int i = 0, j = 0; i < text.length(); i++) {

char c = text.charAt(i);

if (c < 'A' || c > 'Z') {

continue;

}

res += (char) ((c + key.charAt(j) - 2 \* 'A') % 26 + 'A');

j = ++j % key.length();

}

return res;

}

static String decode(String text, final String key) {

String res = "";

text = text.toUpperCase();

for (int i = 0, j = 0; i < text.length(); i++) {

char c = text.charAt(i);

if (c < 'A' || c > 'Z') {

continue;

}

res += (char) ((c - key.charAt(j) + 26) % 26 + 'A');

j = ++j % key.length();

}

return res;

}

public static void main(String[] args) throws java.lang.Exception {

String key = "VIGENERECIPHER";

String msg = "Environment";

System.out.println("Simulating Vigenere Cipher\n------------------------");

System.out.println("Input Message : " + msg);

String enc = encode(msg, key);

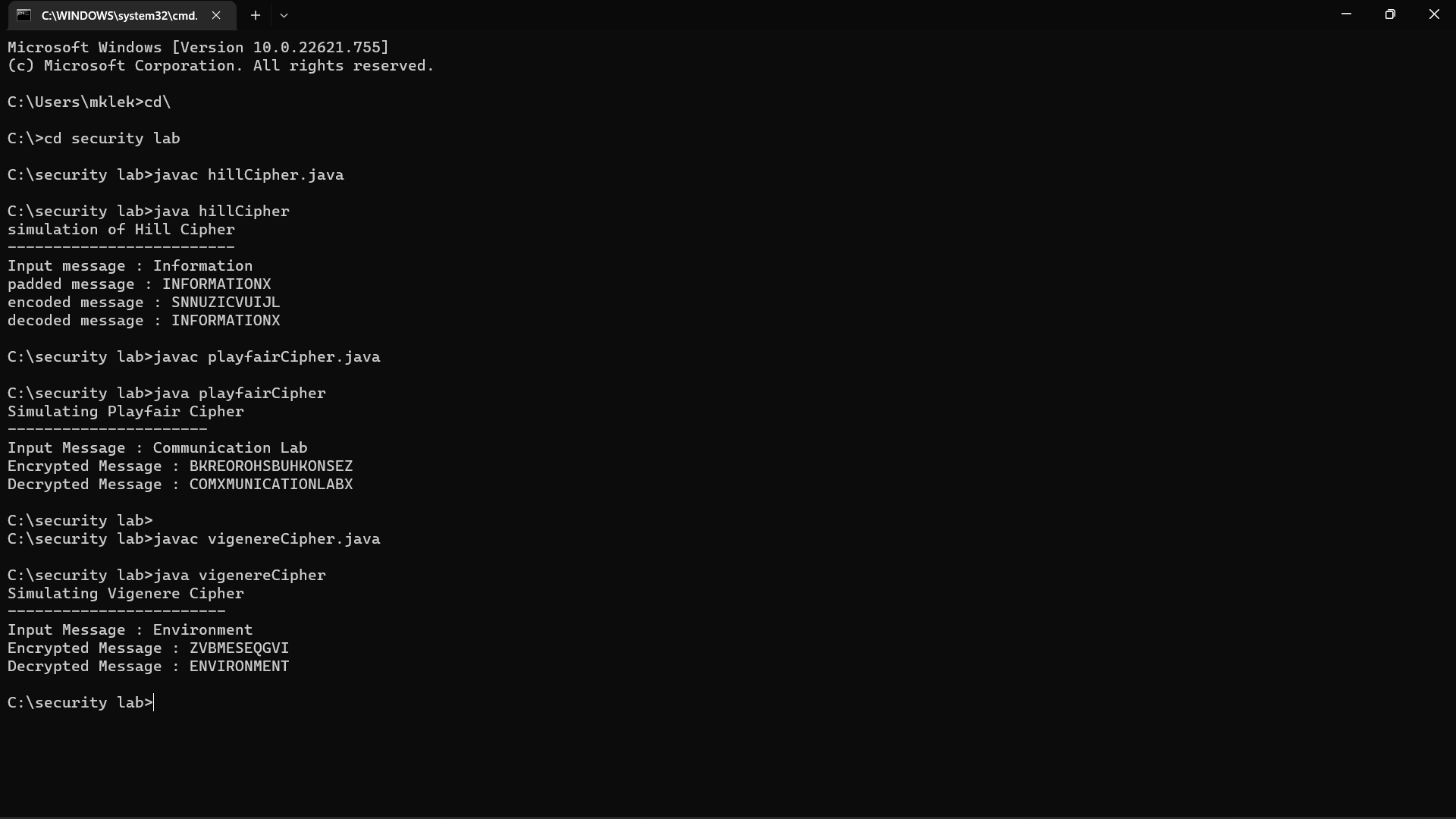
System.out.println("Encrypted Message : " + enc);

System.out.println("Decrypted Message : " + decode(enc, key));

}

}

OUTPUT:



**2(a).Rail Fence Cipher Transposition Technique**

Abdul Rahman S - 7881

import java.util.\*;

class RailFenceBasic{

 int depth;

 String Encryption(String plainText,int depth)throws Exception

 {

  int r=depth,len=plainText.length();

  int c=len/depth;

  char mat[][]=new char[r][c];

  int k=0;

  String cipherText="";

  for(int i=0;i< c;i++)

  {

   for(int j=0;j< r;j++)

   {

    if(k!=len)

     mat[j][i]=plainText.charAt(k++);

    else

     mat[j][i]='X';

   }

  }

  for(int i=0;i< r;i++)

  {

   for(int j=0;j< c;j++)

   {

    cipherText+=mat[i][j];

   }

  }

  return cipherText;

 }

 String Decryption(String cipherText,int depth)throws Exception

 {

  int r=depth,len=cipherText.length();

  int c=len/depth;

  char mat[][]=new char[r][c];

  int k=0;

  String plainText="";

  for(int i=0;i< r;i++)

  {

   for(int j=0;j< c;j++)

   {

    mat[i][j]=cipherText.charAt(k++);

   }

  }

  for(int i=0;i< c;i++)

  {

   for(int j=0;j< r;j++)

   {

    plainText+=mat[j][i];

   }

  }

  return plainText;

 }

}

class RailFence{

 public static void main(String args[])throws Exception

 {

  RailFenceBasic rf=new RailFenceBasic();

                Scanner scn=new Scanner(System.in);

                int depth;

                String plainText,cipherText,decryptedText;

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

                plainText=scn.nextLine();

                System.out.println("Enter depth for Encryption:");

                depth=scn.nextInt();

  cipherText=rf.Encryption(plainText,depth);

  System.out.println("Encrypted text is:\n"+cipherText);

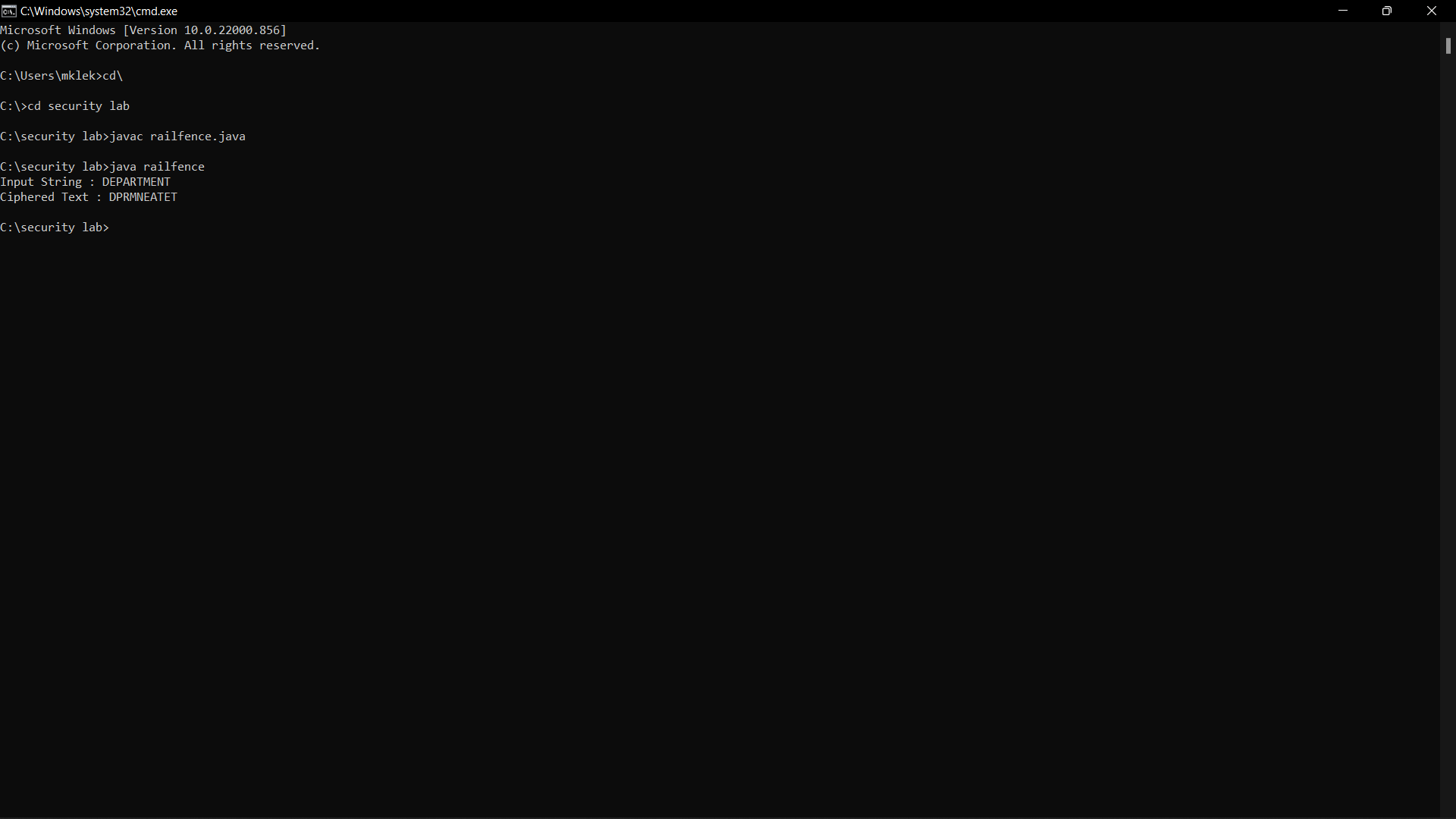
                decryptedText=rf.Decryption(cipherText, depth);

  System.out.println("Decrypted text is:\n"+decryptedText);

 }

}

OUTPUT:



**2(b).Row and Column Transformation Technique**

Abdul Rahman S - 7881

import java.util.\*;

class TransCipher {

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

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

String pl = sc.nextLine();

sc.close();

String s = "";

int start = 0;

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

if (pl.charAt(i) == ' ') {

s = s + pl.substring(start, i);

start = i + 1;

}

}

s = s + pl.substring(start);

System.out.print(s);

System.out.println();

int k = s.length();

int l = 0;

int col = 4;

int row = s.length() / col;

char ch[][] = new char[row][col];

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

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

if (l < k) {

ch[i][j] = s.charAt(l);

l++;

} else {

ch[i][j] = '#';

}

}

}

char trans[][] = new char[col][row];

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

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

trans[j][i] = ch[i][j];

}

}

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

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

System.out.print(trans[i][j]);

}

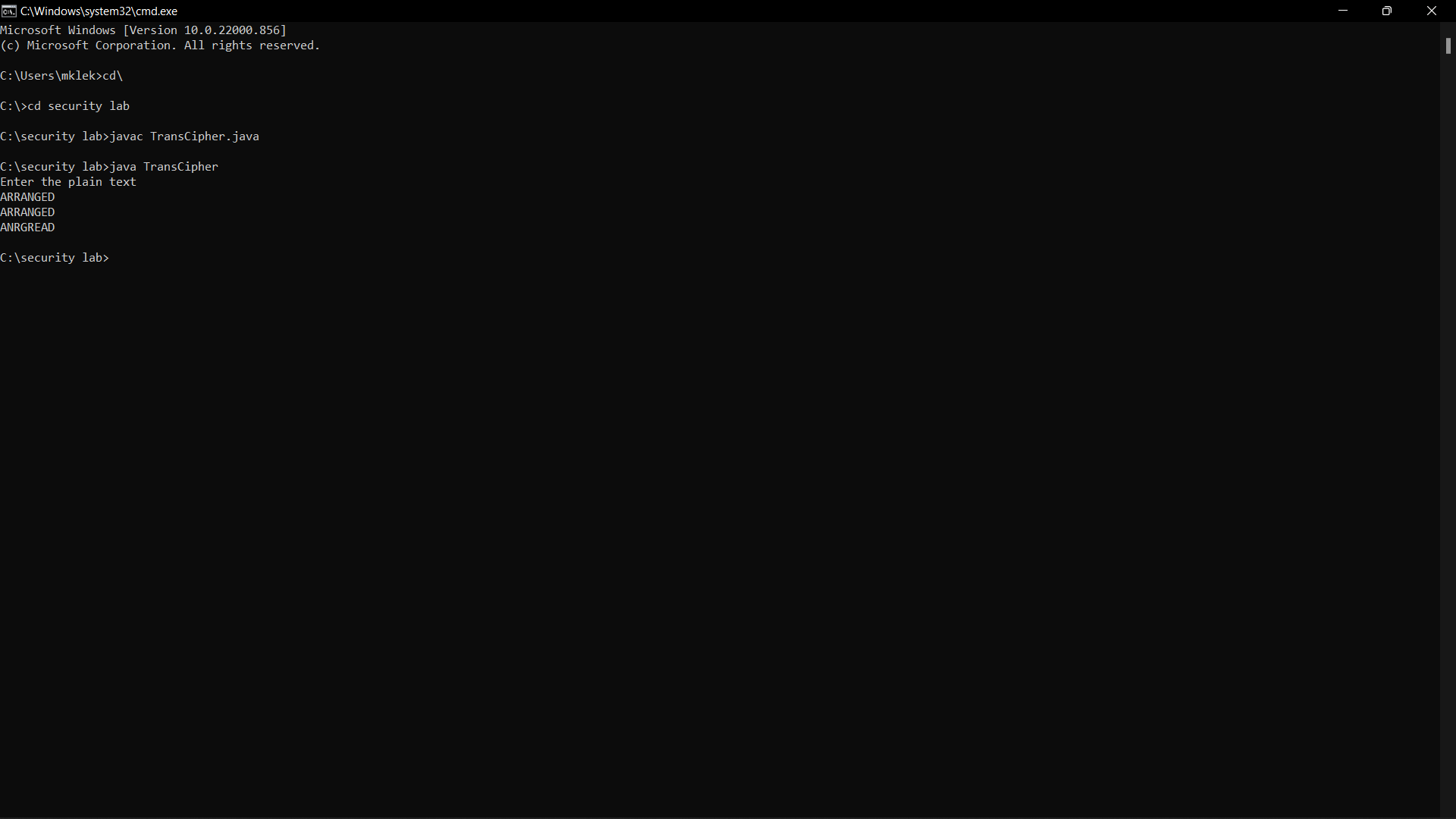
}

System.out.println();

}

}

OUTPUT:



**3.Data Encryption Standard (DES) Algorithm (User Message Encryption )**

Abdul Rahman S - 7881

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;

import javax.crypto.IllegalBlockSizeException;

import javax.crypto.KeyGenerator;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.SecretKey;

public class DES

{

public static void main(String[] argv) {

try{

System.out.println("Message Encryption Using DES Algorithm\n-------");

KeyGenerator keygenerator = KeyGenerator.getInstance("DES");

SecretKey myDesKey = keygenerator.generateKey();

Cipher desCipher;

desCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");

desCipher.init(Cipher.ENCRYPT\_MODE, myDesKey);

byte[] text = "Secret Information ".getBytes();

System.out.println("Message [Byte Format] : " + text);

System.out.println("Message : " + new String(text));

byte[] textEncrypted = desCipher.doFinal(text);

System.out.println("Encrypted Message: " + textEncrypted);

desCipher.init(Cipher.DECRYPT\_MODE, myDesKey);

byte[] textDecrypted = desCipher.doFinal(textEncrypted);

System.out.println("Decrypted Message: " + new

String(textDecrypted));

}catch(NoSuchAlgorithmException e){

e.printStackTrace();

}catch(NoSuchPaddingException e){

e.printStackTrace();

}catch(InvalidKeyException e){

e.printStackTrace();

}catch(IllegalBlockSizeException e){

e.printStackTrace();

}catch(BadPaddingException e){

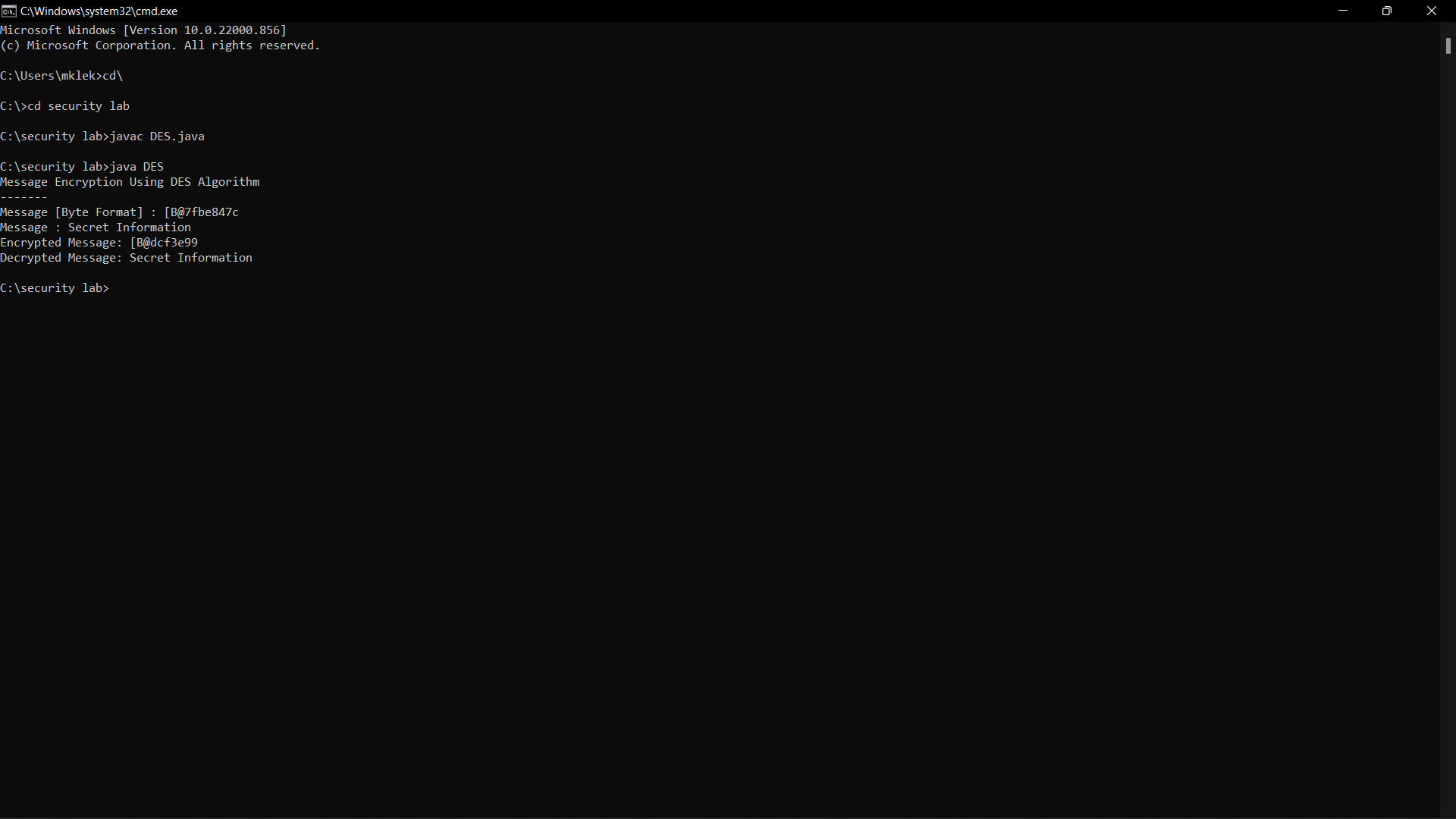
e.printStackTrace();

}

}

}

OUTPUT:



**4. Advanced Encryption Standard (DES) Algorithm ( URL Encryption )**

Abdul Rahman S - 7881

import java.io.UnsupportedEncodingException;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Arrays;

import java.util.Base64;

import javax.crypto.Cipher;

import javax.crypto.spec.SecretKeySpec;

public class AES {

private static SecretKeySpec secretKey;

private static byte[] key;

public static void setKey(String myKey) {

MessageDigest sha = null;

try {

key = myKey.getBytes("UTF-8");

sha = MessageDigest.getInstance("SHA-1");

key = sha.digest(key);

key = Arrays.copyOf(key, 16);

secretKey = new SecretKeySpec(key, "AES");

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

} catch (UnsupportedEncodingException e) {

e.printStackTrace();

}

}

public static String encrypt(String strToEncrypt, String secret) {

try {

setKey(secret);

Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");

cipher.init(Cipher.ENCRYPT\_MODE, secretKey);

return

Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));

} catch (Exception e) {

System.out.println("Error while encrypting: " + e.toString());

}

return null;

}

public static String decrypt(String strToDecrypt, String secret) {

try {

setKey(secret);

Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");

cipher.init(Cipher.DECRYPT\_MODE, secretKey);

return new

String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));

} catch (Exception e) {

System.out.println("Error while decrypting: " + e.toString());

}

return null;

}

public static void main(String[] args) {

final String secretKey = "annaUniversity";

String originalString = "www.annauniv.edu";

String encryptedString = AES.encrypt(originalString, secretKey);

String decryptedString = AES.decrypt(encryptedString, secretKey);

System.out.println("URL Encryption Using AES Algorithm\n------------");

System.out.println("Original URL : " + originalString);

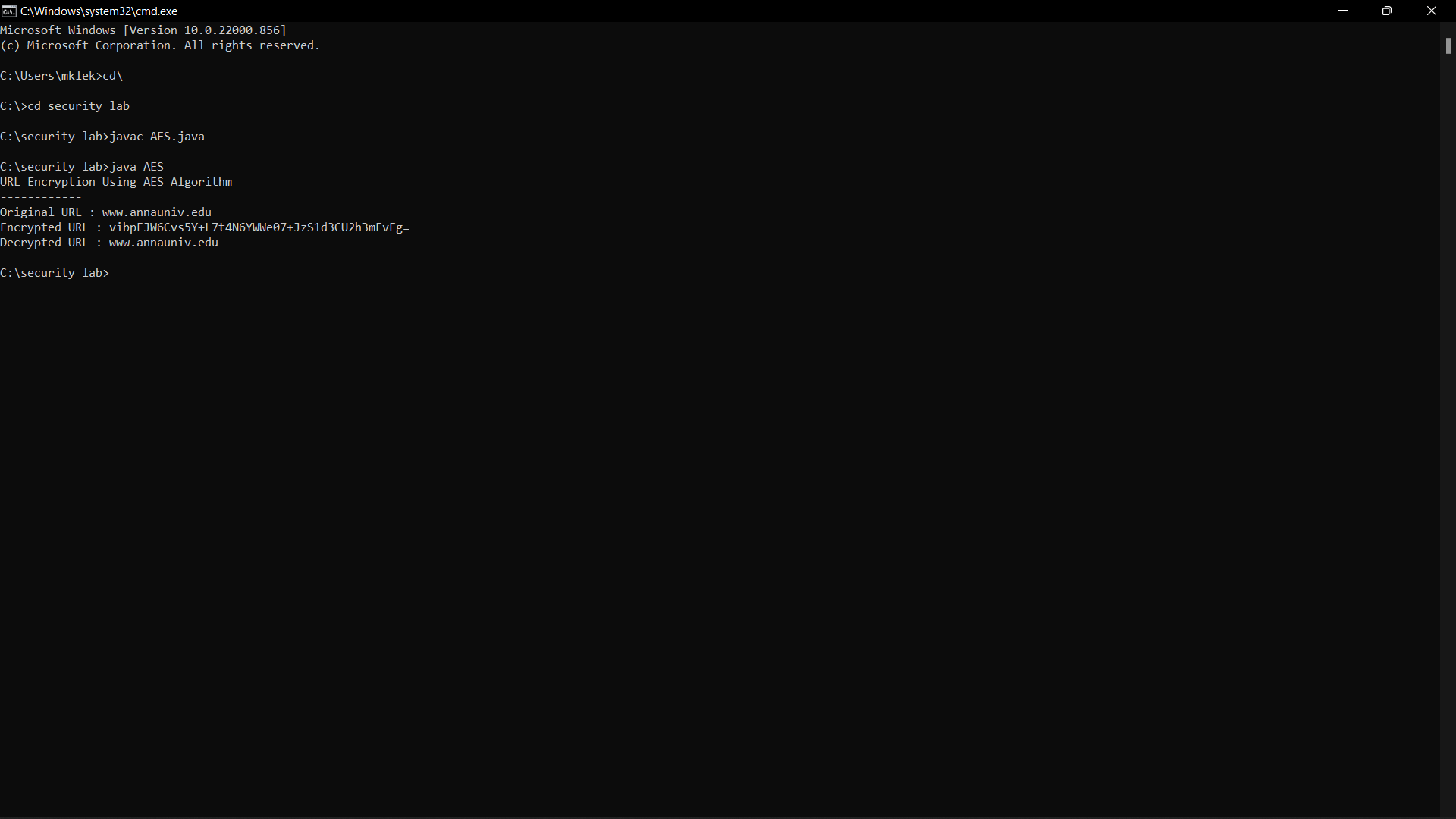
System.out.println("Encrypted URL : " + encryptedString);

System.out.println("Decrypted URL : " + decryptedString);

}

}

OUTPUT:



**5. RSA Algorithm**

Abdul Rahman S - 7881

<html>

<head>

<title>RSA Encryption</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<center>

<h1>RSA Algorithm</h1>

<h2>Implemented Using HTML & Javascript</h2>

<hr>

<table>

<tr>

<td>Enter First Prime Number:</td>

<td><input type="number" value="53" id="p"></td>

</tr>

<tr>

<td>Enter Second Prime Number:</td>

<td><input type="number" value="59" id="q"></p>

</td>

</tr>

<tr>

<td>Enter the Message(cipher text):<br>[A=1, B=2,...]</td>

<td><input type="number" value="89" id="msg"></p>

</td>

</tr>

<tr>

<td>Public Key:</td>

<td>

<p id="publickey"></p>

</td>

</tr>

<tr>

<td>Exponent:</td>

<td>

<p id="exponent"></p>

</td>

</tr>

<tr>

<td>Private Key:</td>

<td>

<p id="privatekey"></p>

</td>

</tr>

<tr>

<td>Cipher Text:</td>

<td>

<p id="ciphertext"></p>

</td>

</tr>

<tr>

<td><button onclick="RSA();">Apply RSA</button></td>

</tr>

</table>

</center>

</body>

<script type="text/javascript">

function RSA() {

var gcd, p, q, no, n, t, e, i, x;

gcd = function (a, b) { return (!b) ? a : gcd(b, a % b); };

p = document.getElementById('p').value;

q = document.getElementById('q').value;

no = document.getElementById('msg').value;

n = p \* q;

t = (p - 1) \* (q - 1);

for (e = 2; e < t; e++) {

if (gcd(e, t) == 1) {

break;

}

}

for (i = 0; i < 10; i++) {

x = 1 + i \* t

if (x % e == 0) {

d = x / e;

break;

}

}

ctt = Math.pow(no, e).toFixed(0);

ct = ctt % n;

dtt = Math.pow(ct, d).toFixed(0);

dt = dtt % n;

document.getElementById('publickey').innerHTML = n;

document.getElementById('exponent').innerHTML = e;

document.getElementById('privatekey').innerHTML = d;

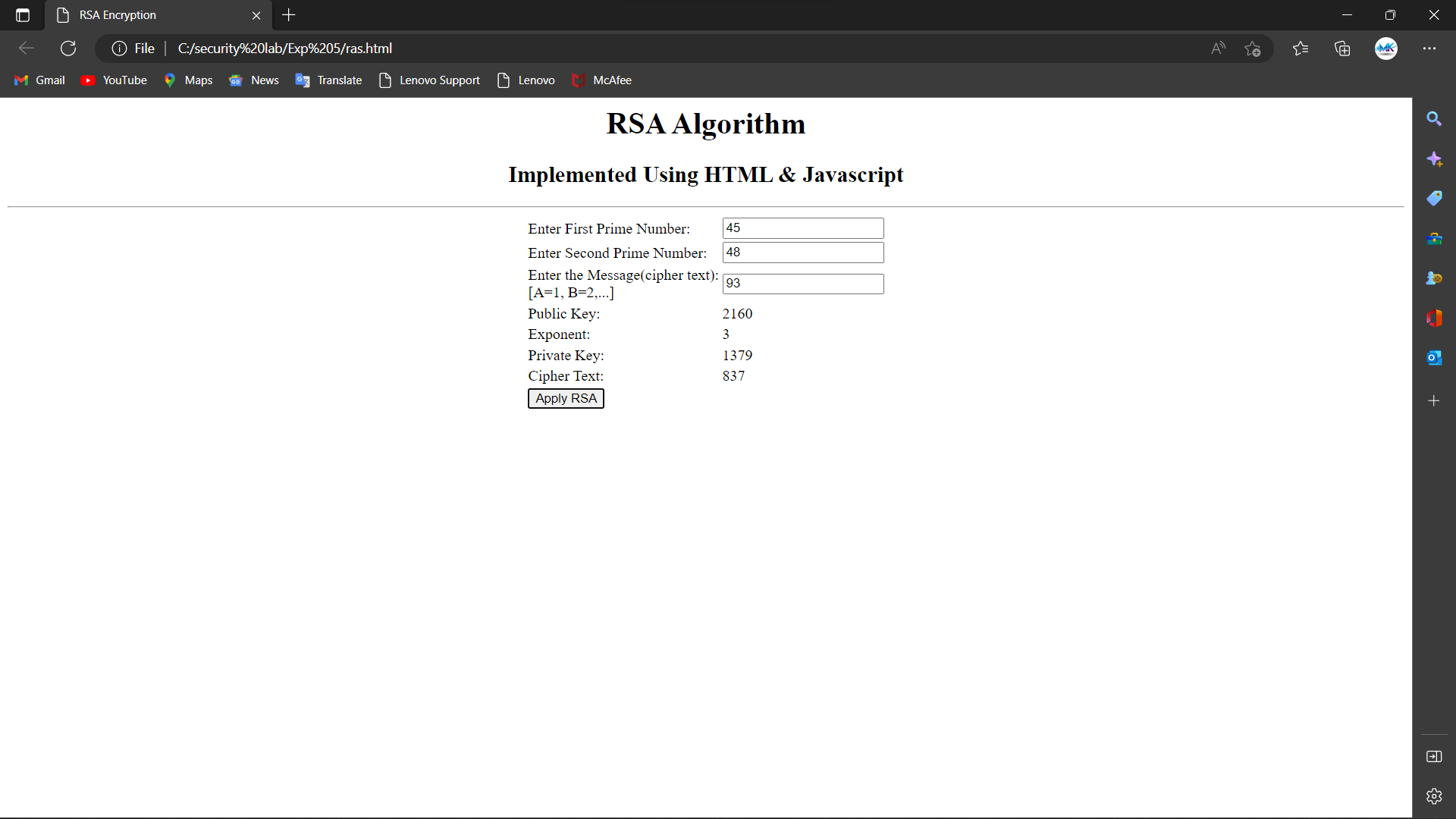
document.getElementById('ciphertext').innerHTML = ct;

}

</script>

</html>

OUTPUT:



**6 . Diffie-Hellman key exchange algorithm**

(S. Manoj Kumar-7577)

import java.util.\*;

class DiffieHellmanAlgorithmExample {

public static void main(String[] args)

{

long P, G, x, a, y, b, ka, kb;

Scanner sc = new Scanner(System.in);

System.out.println("Both the users should be agreed upon the public keys G and P");

System.out.println("Enter value for public key G:");

G = sc.nextLong();

System.out.println("Enter value for public key P:");

P = sc.nextLong();

System.out.println("Enter value for private key a selected by user1:");

a = sc.nextLong();

System.out.println("Enter value for private key b selected by user2:");

b = sc.nextLong();

x = calculatePower(G, a, P);

y = calculatePower(G, b, P);

ka = calculatePower(y, a, P);

kb = calculatePower(x, b, P);

System.out.println("Secret key for User1 is:" + ka);

System.out.println("Secret key for User2 is:" + kb);

}

private static long calculatePower(long x, long y, long P)

{

long result = 0;

if (y == 1){

return x;

}

else{

result = ((long)Math.pow(x, y)) % P;

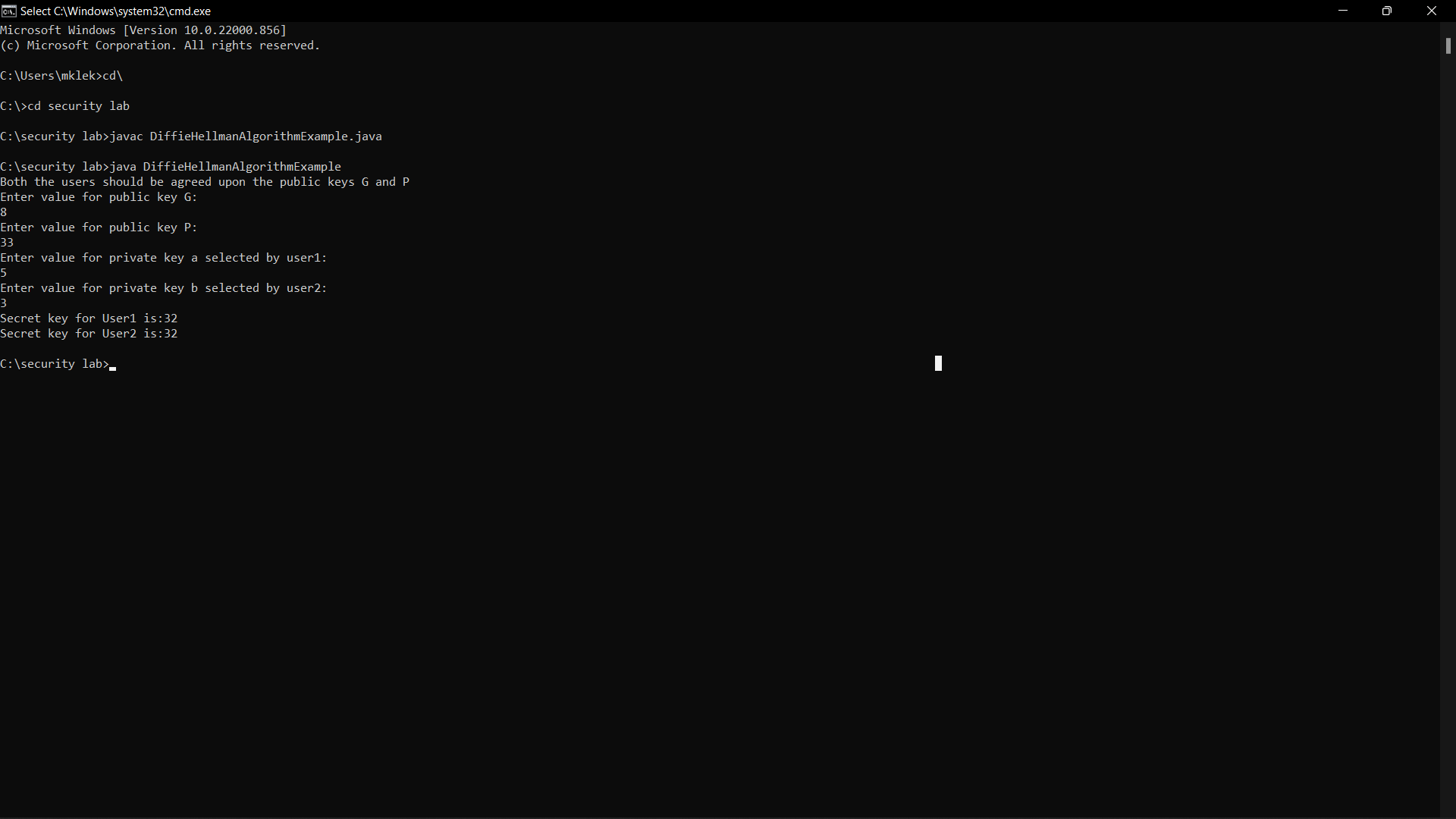
return result;

}

}

}

OUTPUT:



**7. SHA-1 Algorithm**

Abdul Rahman S - 7881

import java.security.\*;

public class sha {

public static void main(String[] a) {

try {

MessageDigest md = MessageDigest.getInstance("SHA1");

System.out.println("Message digest object info:\n-----------------");

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 = "Jazz Music Night";

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);

}

}

private 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 (byte aB : b) {

buf.append(hexDigit[(aB >> 4) & 0x0f]);

buf.append(hexDigit[aB & 0x0f]);

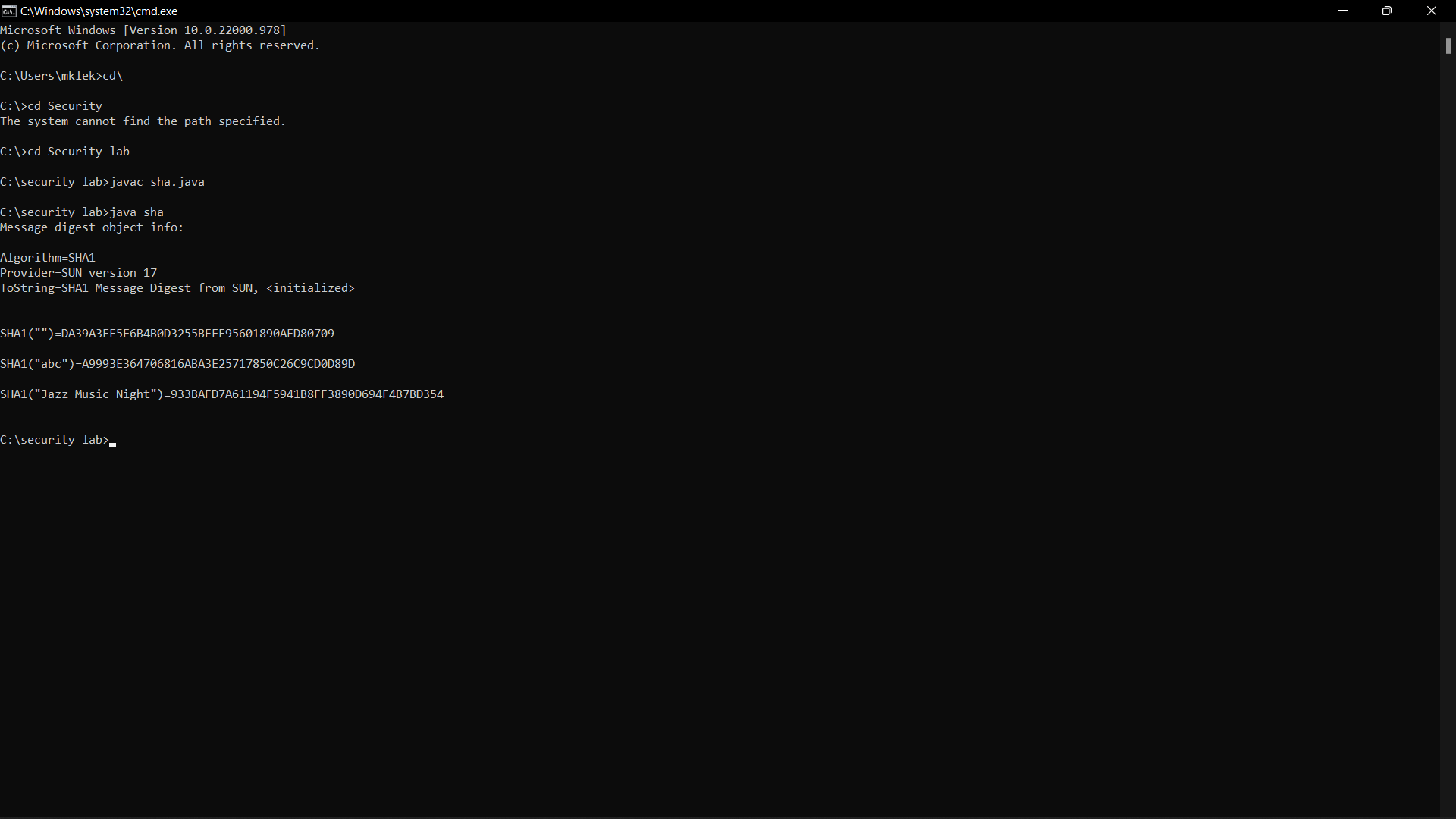
}

return buf.toString();

}

}

OUTPUT:



**8. Digital Signature Standard**

Abdul Rahman S - 7881

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.PrivateKey;

import java.security.Signature;

import java.util.Scanner;

public class CreatingDigitalSignature {

public static void main(String args[]) throws Exception {

Scanner sc = new Scanner(System.in);

System.out.println("Enter some text");

String msg = sc.nextLine();

KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("DSA");

keyPairGen.initialize(2048);

KeyPair pair = keyPairGen.generateKeyPair();

PrivateKey privKey = pair.getPrivate();

Signature sign = Signature.getInstance("SHA256withDSA");

sign.initSign(privKey);

byte[] bytes = "msg".getBytes();

sign.update(bytes);

byte[] signature = sign.sign();

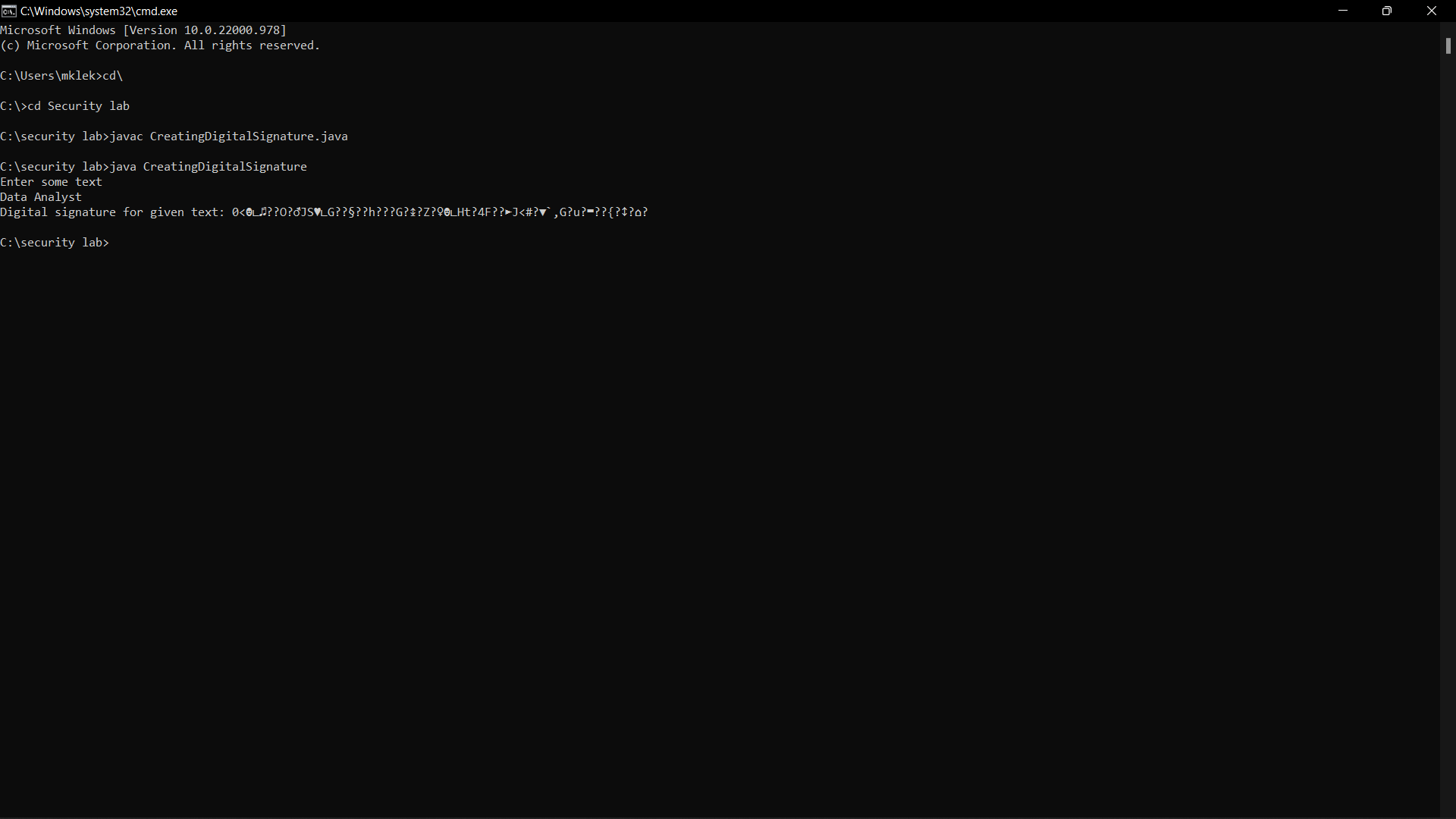
System.out.println("Digital signature for given text: "+new String(signature,

"UTF8"));

}

}

OUTPUT:



**9. Demonstration of Intrusion Detection System(IDS)**

Abdul Rahman S - 7881

1. Download Snort from the Snort.org website.

2. Download Rules. You must register to get the rules.

3. Double-click on the .exe to install snort.

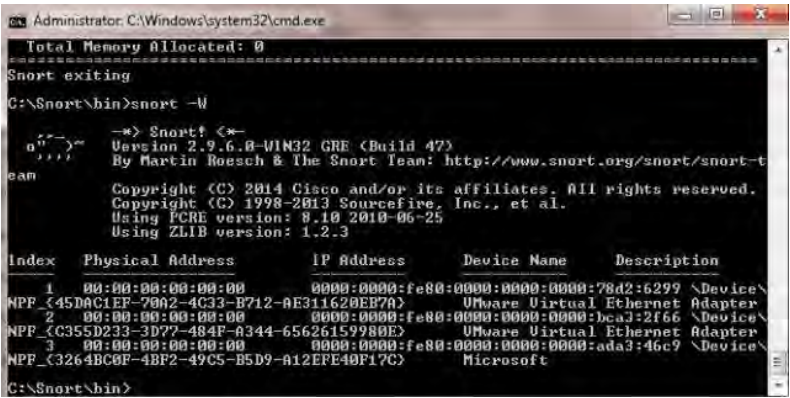
4. Extract the Rules file. You will need WinRAR for the .gz file.

5. Copy all files from the “rules” folder of the extracted folder. Now paste the rules into the “C:\Snort\rules” folder.

6. Copy the “snort. conf” file from the “etc” folder of the extracted folder. You must paste it into the “C:\Snort\etc” folder.

7. Open a command prompt (cmd.exe) and navigate to the folder “C:\Snort\bin” folder. ( at the Prompt, type cd\snort\bin)

8. To start (execute) snort in sniffer mode use the following command: snort -dev -i 3 -i Indicates the interface number.



9. To run snort in IDS mode, you will need to configure the file “snort. conf” according to your network environment.

10. To specify the network address that you want to protect in snort.conf file, look for the following line.

11. To set the addresses of DNS\_SERVERS if you have some on your network.

12. Change the RULE\_PATH variable to the path of the rules folder. var RULE\_PATH c:\snort\rules.

13. Change the path of all library files with the name and path on your system. and you must change the path of snort\_dynamicpreprocessorvariable.

14. Change the path of the “dynamic engine” variable value in the “snort. conf” file.

15 Add the paths for the “include classification. config” and “include reference. config” files.

16. Remove the comment (#) on the line to allow ICMP rules, if it is commented with a #.

17. The comment of the ICMP-info rules comment if it is commented.

18. To add log files to store alerts generated by snort, search for the “output log” test in snort. conf and add the following line:

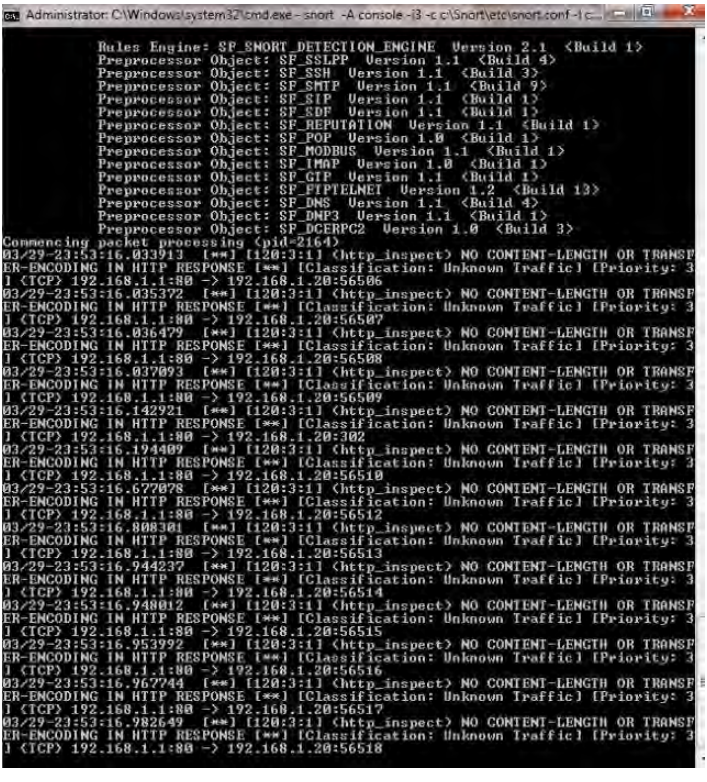
19. Comment (add a #) the whitelist $WHITE\_LIST\_PATH/white\_list.rules and the blacklist

20. Comment out (#) the following lines:

21. Save the “snort. conf” file.

22. To start snort in IDS mode, run the following command:

23. Scan the computer running snort from another computer using PING or NMap (ZenMap).



**10. Exploring N-Stalker, a Vulnerability Assessment Tool**

Abdul Rahman S - 7881

1. Start N-Stalker from a Windows computer. The program is installed under Start ➪ Programs ➪ N-Stalker ➪ N-Stalker Free Edition.

2. Enter a host address or a range of addresses to scan.

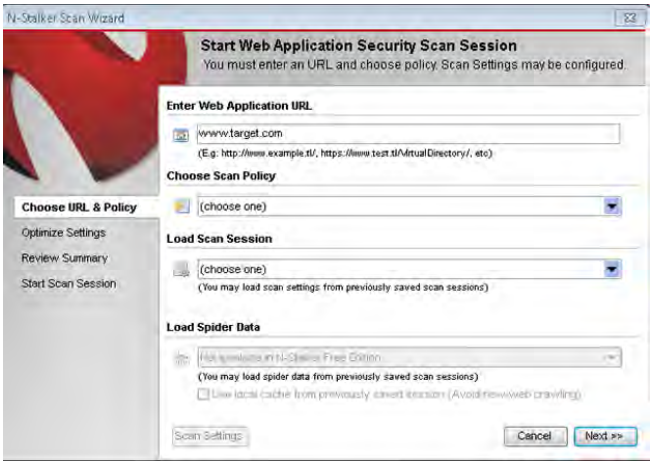
3. Click Start Scan.

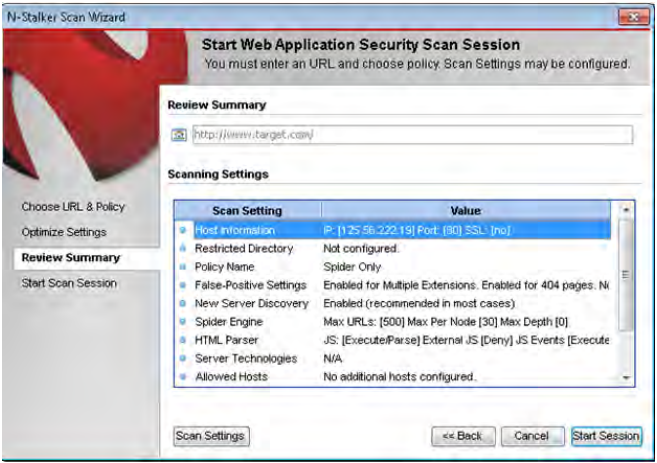
4. After the scan completes, the N-Stalker Report Manager will prompt

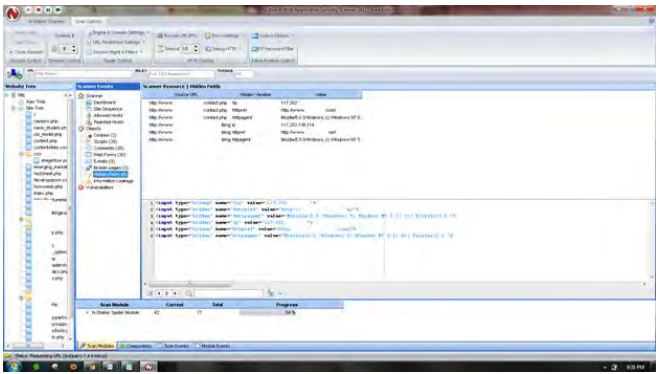
5. you to select a format for the resulting report and choose to Generate HTML.

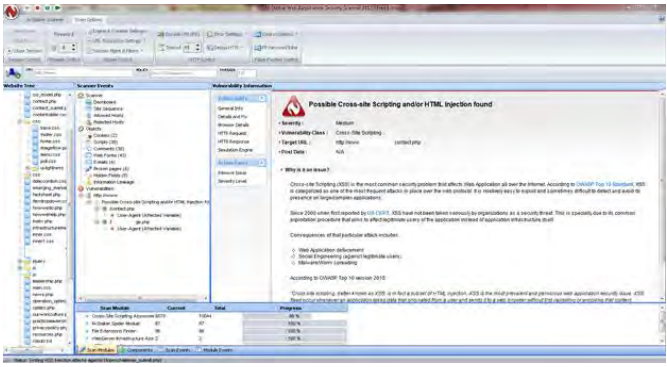
6. Review the HTML report for vulnerabilities.











**11(a) Defeating Malware - Building Trojans**

Abdul Rahman S - 7881

TROJAN:

* In computing, a Trojan horse,or trojan, is any malware which misleads users of its true intent.
* Trojans are generally spread by some form of social engineering, for example where a user is duped into executing an email attachment disguised to appear not suspicious, (e.g., a routine form to be filled in), or by clicking on some fake advertisement on social media or anywhere else.
* Although their payload can be anything, many modern forms act as a backdoor, contacting a controller which can then have unauthorized access to the affected computer.
* Trojans may allow an attacker to access users' personal information such as banking information, passwords, or personal identity.
* **Example:** Ransomware attacks are often carried out using a trojan.

CODE:

Trojan.bat

@echo off

:x

start mspaint

start notepad

start cmd

start explorer

start control

start calc

goto x

OUTPUT:

(MS-Paint, Notepad, Command Prompt, Explorer will open infinitely)

**11(b) Defeating Malware - Rootkit hunter**

Abdul Rahman S - 7881

