1.Write a C program that contains a String (char) with a value hello world. The program should XOR each character in this string with 0 and display the result.

Program:

#include <stdio.h>

#include <string.h>

#include <math.h>

void main(){

char str[]="hello world";

int i;

int len=strlen(str);

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

char result=str[i]^0;

printf("%c",result);

}

}

Output:

hello world

1. Write a C program that contains a String (char) with a value hello world. The program should AND and XOR each character in this string with 127 and display the result.

Program:

#include <stdio.h>

#include <string.h>

#include <math.h>

void main(){

char str[]="hello world";

int i;

int len=strlen(str);

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

char and=str[i]&127;

char or=str[i]&127;

char xor=str[i]^127;

printf("and:%c\t, or:%c\t,xor:%c\n",and,or,xor );

}

}

Output:

and:h , or:h ,xor:

and:e , or:e ,xor:

and:l , or:l ,xor:

and:l , or:l ,xor:

and:o , or:o ,xor:\_x0010\_

and: , or: ,xor:\_

and:w , or:w ,xor:

and:o , or:o ,xor:\_x0010\_

and:r , or:r ,xor:

and:l , or:l ,xor:

and:d , or:d ,xor:

1. a) Write a java program to implement Caesar cipher.

Program:

import java.util.\*;

public class CaesarCipher {

public static String encrypt(String plaintext, int shift) {

StringBuilder encryptedText = new StringBuilder();

for (char character : plaintext.toCharArray()) {

if (Character.isLetter(character)) {

char base = Character.isLowerCase(character) ? 'a' : 'A';

int offset = character - base;

char encryptedChar = (char) ((offset + shift) % 26 + base);

encryptedText.append(encryptedChar);

} else {

encryptedText.append(character);

}

}

return encryptedText.toString();

}

public static String decrypt(String ciphertext, int shift) {

return encrypt(ciphertext, 26 - shift); // Decryption is just encryption with the inverse shift

}

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

System.out.println("enter the plain text:");

String plaintext = sc.nextLine();

int shift = 3;

String encryptedText = encrypt(plaintext, shift);

String decryptedText = decrypt(encryptedText, shift);

System.out.println("Original Text: " + plaintext);

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

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

}

}

Output:

enter the plain text:

hello

Original Text: hello

Encrypted Text: khoor

Decrypted Text: hello

3.b)Write a java program to implement substitution cypher.

Program:

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 a = "abcdefghijklmnopqrstuvwxyz";

String b = "zyxwvutsrqponmlkjihgfedcba";

System.out.print("Enter any string: ");

String str = br.readLine();

String decrypt = "";

char c;

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

c = str.charAt(i);

int j = a.indexOf(c);

decrypt += b.charAt(j);

}

System.out.println("The encrypted data is: " + decrypt);

}

}

Output:

Enter any string: hello

The encrypted data is: svool

3.c)Write a java program to implement hill cypher.

import java.util.\*;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class HillCipherExample

{

int[] l\_m;

int[][] k\_m;

int[] r\_m;

static int ch;

int [][] nk;

public void perf\_Division(String t, int str)

{

while (t.length() > str)

{

String l = t.substring(0, str);

t = t.substring(str, t.length());

calLineMatrix(l);

if(ch ==1)

{

multiplyLineByKey(l.length());

}

else

{

multiplyLineByInvKey(l.length());

}

showResult(l.length());

}

if (t.length() == str)

{

if(ch ==1)

{

calLineMatrix(t);

multiplyLineByKey(t.length());

showResult(t.length());

}

else

{

calLineMatrix(t);

this.multiplyLineByInvKey(t.length());

showResult(t.length());

}

}

else if (t.length() < str)

{

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

t = t + 'x';

if(ch ==1)

{

calLineMatrix(t);

multiplyLineByKey(t.length());

showResult(t.length());

}

else

{

calLineMatrix(t);

multiplyLineByInvKey(t.length());

showResult(t.length());

}

}

}

public void calKeyMatrix(String key, int len)

{

k\_m = new int[len][len];

int k = 0;

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

{

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

{

k\_m[i][j] = ((int) key.charAt(k)) - 97;

k++;

}

}

}

public void calLineMatrix(String l)

{

l\_m = new int[l.length()];

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

{

l\_m[i] = ((int) l.charAt(i)) - 97;

}

}

public void multiplyLineByKey(int len)

{

r\_m = new int[len];

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

{

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

{

r\_m[i] += k\_m[i][j] \* l\_m[j];

}

r\_m[i] %= 26;

}

}

public void multiplyLineByInvKey(int len)

{

r\_m = new int[len];

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

{

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

{

r\_m[i] += nk[i][j] \* l\_m[j];

}

r\_m[i] %= 26;

}

}

public void showResult(int len)

{

String result = "";

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

{

result += (char) (r\_m[i] + 97);

}

System.out.print(result);

}

public int calDeter\_minant(int A[][], int N)

{

int resultOfDet;

switch (N)

{

case 1: resultOfDet = A[0][0];

break;

case 2: resultOfDet = A[0][0] \* A[1][1] - A[1][0] \* A[0][1];

break;

default: resultOfDet = 0;

for (int j1 = 0; j1 < N; j1++)

{

int m[][] = new int[N - 1][N - 1];

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

{

int j2 = 0;

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

{

if (j == j1)

continue;

m[i - 1][j2] = A[i][j];

j2++;

}

}

resultOfDet += Math.pow(-1.0, 1.0 + j1 + 1.0) \* A[0][j1]\* calDeter\_minant(m, N - 1);

}

break;

}

return resultOfDet;

}

public void cofact(int num[][], int f)

{

int b[][], fac[][];

b = new int[f][f];

fac = new int[f][f];

int p, q, m, n, i, j;

for (q = 0; q < f; q++)

{

for (p = 0; p < f; p++)

{

m = 0;

n = 0;

for (i = 0; i < f; i++)

{

for (j = 0; j < f; j++)

{

b[i][j] = 0;

if (i != q && j != p)

{

b[m][n] = num[i][j];

if (n < (f - 2))

n++;

else

{

n = 0;

m++;

}

}

}

}

fac[q][p] = (int) Math.pow(-1, q + p) \* calDeter\_minant(b, f - 1);

}

}

trans(fac, f);

}

void trans(int fac[][], int r)

{

int i, j;

int b[][], inv[][];

b = new int[r][r];

inv = new int[r][r];

int d = calDeter\_minant(k\_m, r);

int mi = mi(d % 26);

mi %= 26;

if (mi < 0)

mi += 26;

for (i = 0; i < r; i++)

{

for (j = 0; j < r; j++)

{

b[i][j] = fac[j][i];

}

}

for (i = 0; i < r; i++)

{

for (j = 0; j < r; j++)

{

inv[i][j] = b[i][j] % 26;

if (inv[i][j] < 0)

inv[i][j] += 26;

inv[i][j] \*= mi;

inv[i][j] %= 26;

}

}

nk = inv;

}

public int mi(int d)

{

int q, r1, r2, r, t1, t2, t;

r1 = 26;

r2 = d;

t1 = 0;

t2 = 1;

while (r1 != 1 && r2 != 0)

{

q = r1 / r2;

r = r1 % r2;

t = t1 - (t2 \* q);

r1 = r2;

r2 = r;

t1 = t2;

t2 = t;

}

return (t1 + t2);

}

public void matrixtoinvkey(int inv[][], int n)

{

String invkey = "";

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

{

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

{

invkey += (char) (inv[i][j] + 97);

}

}

System.out.print(invkey);

}

public boolean check(String key, int len)

{

calKeyMatrix(key, len);

int d = calDeter\_minant(k\_m, len);

d = d % 26;

if (d == 0)

{

System.out.println("Key is not invertible");

return false;

}

else if (d % 2 == 0 || d % 13 == 0)

{

System.out.println("Key is not invertible");

return false;

}

else

{

return true;

}

}

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

{

HillCipherExample obj = new HillCipherExample();

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Menu:\n1: Encryption\n2: Decryption");

ch = Integer.parseInt(in.readLine());

System.out.println("Enter the line: ");

String l = in.readLine();

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

String key = in.readLine();

double sq = Math.sqrt(key.length());

if (sq != (long) sq)

System.out.println("Cannot For\_m a square matrix");

else

{

int size = (int) sq;

if (obj.check(key, size))

{

System.out.println("Result:");

obj.cofact(obj.k\_m, size);

obj.perf\_Division(l, size);

}

}

}

}

Output:

Menu:

1: Encryption

2: Decryption

1

Enter the line:

pree

Enter the key:

hill

Result:

hoik

Menu:

1: Encryption

2: Decryption

2

Enter the line:

hoik

Enter the key:

hill

Result:

Pree

Menu:

1: Encryption

2: Decryption

1

Enter the line:

pree

Enter the key:

hain

Key is not invertible

4.Write a java program to implement DES Algorithm.

import java.util.\*;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.security.spec.KeySpec;

import javax.crypto.Cipher;

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.DESedeKeySpec;

import sun.misc.BASE64Decoder;

import sun.misc.BASE64Encoder;

public class DES

{

private static final String UNICODE\_FORMAT = "UTF8";

public static final String DESEDE\_ENCRYPTION\_SCHEME = "DESede";

privateKeySpec myKeySpec;

privateSecretKeyFactory mySecretKeyFactory;

private Cipher cipher;

byte[] keyAsBytes;

private String myEncryptionKey;

private String myEncryptionScheme;

SecretKey key;

static BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); public DES() throws Exception

{

// TODO code application logic here

myEncryptionKey = "ThisIsSecretEncryptionKey";

myEncryptionScheme =DESEDE\_ENCRYPTION\_SCHEME;

keyAsBytes = myEncryptionKey.getBytes(UNICODE\_FORMAT); myKeySpec= new DESedeKeySpec(keyAsBytes);

mySecretKeyFactory = SecretKeyFactory.getInstance(myEncryptionScheme);

cipher = Cipher.getInstance(myEncryptionScheme);

key = mySecretKeyFactory.generateSecret(myKeySpec);

}

public String encrypt(String unencryptedString)

{

String encryptedString = null;

try

{

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

byte[] plainText = unencryptedString.getBytes(UNICODE\_FORMAT);

byte[] encryptedText = cipher.doFinal(plainText);

BASE64Encoder base64encoder = new BASE64Encoder();

encryptedString = base64encoder.encode(encryptedText);

}

catch (Exception e)

{

e.printStackTrace();

}

return encryptedString;

}

public String decrypt(String encryptedString)

{

String decryptedText=null;

try

{

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

BASE64Decoder base64decoder = new BASE64Decoder();

byte[] encryptedText = base64decoder.decodeBuffer(encryptedString);

byte[] plainText = cipher.doFinal(encryptedText);

decryptedText=bytes2String(plainText);

}

catch (Exception e)

{

e.printStackTrace();

}

return decryptedText;

}

private static String bytes2String(byte[] bytes)

{

StringBufferstringBuffer = new StringBuffer();

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

{

stringBuffer.append((char) bytes[i]);

}

return stringBuffer.toString();

}

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

{

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

DES myEncryptor= new DES();

String stringToEncrypt = br.readLine();

String encrypted = myEncryptor.encrypt(stringToEncrypt);

String decrypted = myEncryptor.decrypt(encrypted); System.out.println("\nString To Encrypt: " +stringToEncrypt); System.out.println("\nEncrypted Value : " +encrypted);

System.out.println("\nDecrypted Value : " +decrypted);

System.out.println("");

}

}

Output:

Enter the string: Welcome

String To Encrypt: Welcome

Encrypted Value : BPQMwc0wKvg

Decrypted Value : Welcome

5.Write a C/JAVA program to implement the Rijndael algorithm logic.

import java.security.\*;

import javax.crypto.\*;

import javax.crypto.spec.\*;

import java.io.\*;

public class AES

{

public static String asHex (byte buf[])

{

StringBuffer strbuf = new StringBuffer(buf.length \* 2);

int i;

for (i = 0; i < buf.length; i++)

{ if (((int) buf[i] & 0xff) < 0x10)

strbuf.append("0");

strbuf.append(Long.toString((int) buf[i] & 0xff, 16));

}

return strbuf.toString();

}

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

{

String message="AES still rocks!!";

// Get the KeyGenerator

KeyGenerator kgen = KeyGenerator.getInstance("AES");

kgen.init(128);

// 192 and 256 bits may not be available // Generate the secret key specs.

SecretKey skey = kgen.generateKey();

byte[] raw = skey.getEncoded();

SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");

// Instantiate the cipher

Cipher cipher = Cipher.getInstance("AES");

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

byte[] encrypted = cipher.doFinal((args.length == 0 ? message : args[0]).getBytes());

System.out.println("encrypted string: " + asHex(encrypted));

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

byte[] original = cipher.doFinal(encrypted);

String originalString = new String(original);

System.out.println("Original string: " + originalString + " " + asHex(original));

}

}

Output:

encrypted string: 0595ea453df14e433abe1c0712632d55641af5c25e8c7092b03947163132977e

Original string: AES still rocks!! 414553207374696c6c20726f636b732121

6.Write a java program to implement Blowfish algorithm.

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

import java.util.Scanner;

public class BlowFish {

public static void main(String[] args) {

try {

// Get the KeyGenerator

KeyGenerator kgen = KeyGenerator.getInstance("Blowfish");

Cipher cipher = Cipher.getInstance("Blowfish");

// Generate the secret key

SecretKey skey = kgen.generateKey();

byte[] raw = skey.getEncoded();

SecretKeySpec skeySpec = new SecretKeySpec(raw, "Blowfish");

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

// Input message from the user via command line

Scanner scanner = new Scanner(System.in);

System.out.print("Input your message: ");

String inputText = scanner.nextLine();

byte[] encrypted = cipher.doFinal(inputText.getBytes());

// Decrypt the encrypted message

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

byte[] decrypted = cipher.doFinal(encrypted);

// Display the encrypted and decrypted messages

System.out.println("\nOriginal text: " + inputText +

"\n\nEncrypted text: " + new String(encrypted) +

"\n\nDecrypted text: " + new String(decrypted)

);

scanner.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

OUTPUT:

Input your message:cool

Original text:cool

Encrypted text:.??X?.?

Decrypted text:cool

1. Write a Java Program to implement RSA-Algorithm.

Program:

import java.util.Scanner;

public class RSAEncryption {

public static int modfun(int a, int n, int b) {

if (b == 1) {

return a % n;

} else {

return ((a % n) \* modfun(a, n, b - 1)) % n;

}

}

public static int gcd(int a, int b) {

if (a == 0) {

return b;

} else if (b == 0) {

return a;

} else if (a == b) {

return a;

} else if (a > b) {

return gcd(a - b, b);

} else {

return gcd(a, b - a);

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int a, b, n, e, d = 0, phi, m;

int em, dm;

System.out.println("Enter two prime numbers:");

a = scanner.nextInt();

b = scanner.nextInt();

n = a \* b;

phi = (a-1)\*(b-1);

System.out.println(phi);

System.out.print("Enter e value: ");

e = scanner.nextInt();

if (0 < e && e < n && gcd(e, phi) == 1) {

for (int i = 2; i < n; i++) {

if ((e \* i) % phi == 1) {

System.out.println("d value is: " + i);

d = i;

break;

}

}

System.out.printf("KU {%d %d}\n", e, n);

System.out.printf("KR {%d %d}\n", d, n);

System.out.print("Enter message: ");

m = scanner.nextInt();

em = modfun(m, n, e);

dm = modfun(em, n, d);

System.out.println("Encrypted message: " + em);

System.out.println("Decrypted message: " + (dm % n));

} else {

System.out.println("Invalid e value");

}

}

}

Output:

Enter two prime numbers:

11 17

160

Enter e value: 7

d value is: 23

KU {7 187}

KR {23 187}

Enter message: 21

Encrypted message: 98

Decrypted message: 21

8.Implement Diffie-hellman key exchange mechanism using html & javascript.

Index.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Diffie-Hellman Key Exchange</title>

</head>

<body>

<h1>Diffie-Hellman Key Exchange Example</h1>

<button onclick="startKeyExchange()">Start Key Exchange</button>

<div id="output"></div>

<script src="script.js"></script>

</body>

</html>

Script.js:

const p = 23;

const g = 5;

let privateA, privateB;

let publicA, publicB;

function startKeyExchange() {

privateA = Math.floor(Math.random() \* (p - 1)) + 1;

publicA = (g \*\* privateA) % p;

privateB = Math.floor(Math.random() \* (p - 1)) + 1;

publicB = (g \*\* privateB) % p;

const sharedSecretA = (publicB \*\* privateA) % p;

const sharedSecretB = (publicA \*\* privateB) % p;

document.getElementById('output').innerHTML = `

Private key for party A: ${privateA}<br>

Public key for party A: ${publicA}<br>

Shared secret for party A: ${sharedSecretA}<br><br>

Private key for party B: ${privateB}<br>

Public key for party B: ${publicB}<br>

Shared secret for party B: ${sharedSecretB}

`;

}

Output:

Run index.html

Click the button.the following output is generated(note that the output changes every time the button is pressed):

Diffie-Hellman Key Exchange Example

Private key for party A: 8

Public key for party A: 10

Shared secret for party A: 9

Private key for party B: 7

Public key for party B: 20

Shared secret for party B: 9

9.Calculate the Message digest of a text using SHA-1 algorithm in java.

Program:

import java.security.\*;

import java.util.\*;

import java.security.\*;

import java.util.\*;

class JceSha1Test {

public static void main(String[] a) {

try {

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

String input = "";

md.update(input.getBytes());

byte[] output = md.digest();

System.out.println();

System.out.println("SHA1(\"" + input + "\") =");

System.out.println(" " + bytesToHex(output));

Scanner sc = new Scanner(System.in);

System.out.println("enter the message:");

String str = sc.nextLine();

md.update(str.getBytes());

output = md.digest();

System.out.println();

System.out.println("SHA1(\"" + str + "\") =");

System.out.println(" " + bytesToHex(output));

} 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:

SHA1("") =

DA39A3EE5E6B4B0D3255BFEF95601890AFD80709

enter the message:

hello

SHA1("hello") =

AAF4C61DDCC5E8A2DABEDE0F3B482CD9AEA9434D

10.Calculate the Message digest of a text using MD-5 algorithm In java.

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.\*;

// Java program to calculate MD5 hash value

public class MD5 {

public static String getMd5(String input) {

try {

// Static getInstance method is called with hashing MD5

MessageDigest md = MessageDigest.getInstance("MD5");

// digest() method is called to calculate the message digest

// of an input; digest() returns an array of bytes

byte[] messageDigest = md.digest(input.getBytes());

// Convert the byte array into a signum representation

BigInteger no = new BigInteger(1, messageDigest);

// Convert the message digest into a hex value

String hashtext = no.toString(16);

while (hashtext.length() < 32) {

hashtext = "0" + hashtext;

}

return hashtext;

}

// For specifying wrong message digest algorithms

catch (NoSuchAlgorithmException e) {

throw new RuntimeException(e);

}

}

// Driver code

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

Scanner sc = new Scanner(System.in);

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

String s = sc.nextLine();

System.out.println("Your Hash Code Generated by MD5 is: " + getMd5(s));

}

}

Output:

Enter the plain text:

super

Your Hash Code Generated by MD5 is: 1b3231655cebb7a1f783eddf27d254ca

1. Program to implement rc4 algorithm

Program:

#include <stdio.h>

int main()

{

int i,j,s[10],t[10],k[4],pt[4],temp,p,q,r;

printf("enter plain text:\n");

for(i=0;i<4;i++)

scanf("%d",&pt[i]);

printf("enter key:");

for(i=0;i<4;i++)

scanf("%d",&k[i]);

for(i=0;i<8;i++){

s[i]=i;

t[i]=k[i%4];

}

j=0;

for(i=0;i<8;i++)

{

j=(j+s[i]+t[i])%8;

temp=s[i];

s[i]=s[j];

s[j]=temp;

}

i=j=0;

printf("encrypted text:");

while(i<4){

i=(i+1)%8;

j=(j+s[i])%8;

temp=s[i];

s[i]=s[j];

s[j]=temp;

p=(s[i]+s[j])%8;

q=s[p];

r=q^pt[i-1];

printf("%d",r);

}

printf("\n");

}

Output:

enter plain text:

1

2

2

2

enter key:1

2

3

6

encrypted text:4323

1. Program to implement polyalphabetic cipher

Program:

import java.util.Scanner;

public class VigenereCipher {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String plaintext = sc.nextLine();

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

String key = sc.nextLine();

String ciphertext = encrypt(plaintext, key);

System.out.println("Ciphertext: " + ciphertext);

String decryptedText = decrypt(ciphertext, key);

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

}

public static String encrypt(String plaintext, String key) {

String ciphertext = "";

plaintext = plaintext.toUpperCase();

key = key.toUpperCase();

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

char c = plaintext.charAt(i);

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

continue;

}

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

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

}

return ciphertext;

}

public static String decrypt(String ciphertext, String key) {

String plaintext = "";

ciphertext = ciphertext.toUpperCase();

key = key.toUpperCase();

int j = 0;

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

char c = ciphertext.charAt(i);

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

continue;

}

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

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

}return plaintext;

}

}

output:-

Enter the plaintext:

thebigwheel

Enter the key:

security

Ciphertext: LLGVZOPFWIN

Decrypted text: THEBIGWHEEL

1. Program to implement one time pad cipher

Program:

import java.util.Scanner;

public class OneTimePad {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String plaintext = sc.nextLine();

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

String key = sc.nextLine();

String ciphertext = encrypt(plaintext, key);

System.out.println("Ciphertext: " + ciphertext);

String decryptedText = decrypt(ciphertext, key);

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

}

public static String encrypt(String plaintext, String key) {

String ciphertext = "";

plaintext = plaintext.toUpperCase();

key = key.toUpperCase();

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

char c = plaintext.charAt(i);

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

continue;

}

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

}

return ciphertext;

}

public static String decrypt(String ciphertext, String key) {

String plaintext = "";

ciphertext = ciphertext.toUpperCase();

key = key.toUpperCase();

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

char c = ciphertext.charAt(i);

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

continue;

}

plaintext += (char) ((c - key.charAt(i) + 26) % 26 + 'A');

}

return plaintext;

}

}

output:-

Enter the plaintext:

hello

Enter the key:

axcmp

Ciphertext: HBNXD Decrypted text: HELLO