**Experiment no.:**

**Aim**: **Implement ceaser cipher &simple columnar techniques.**

**Theory:**

**Caesar cipher**

* The Caesar cipher is the simplest and oldest method of cryptography. The Caesar cipher method is based on a mono-alphabetic cipher and is also called a shift cipher or additive cipher. Julius Caesar used the shift cipher (additive cipher) technique to communicate with his officers. For this reason, the shift cipher technique is called the Caesar cipher. The Caesar cipher is a kind of replacement (substitution) cipher, where all letter of plain text is replaced by another letter.

**The formula of encryption is:**

En (x) = (x + n) mod 26

**The formula of decryption is:**

Dn (x) = (xi - n) mod 26

If any case (Dn) value becomes negative (-ve), in this case, we will add 26 in the negative value.

**Where,**

E denotes the encryption  
D denotes the decryption  
x denotes the letters value  
n denotes the key value (shift value)

**Example:**

Here we shift some number of characters based on the given key

Plaintext: Krishna

Key: 8

***Here the key was 8 so first eight characters are shifted /rotated behind.***

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** |

Cipher Text : SZQAPVI

**Algorithm**:

Step 1: Read the plain text from the user

Step 2: Read the key from the user

Step 3: If the key is positive then encrypt the text by adding the key with each character in the plain text

Step 4: Else subtract the key from plain text

Step 5: Display the cipher text obtained above

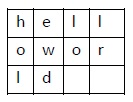
**Simple columnar techniques**

* Transposition Cipher is a cryptographic algorithm where the order of alphabets in the plaintext is rearranged to form a cipher text. In this process, the actual plain text alphabets are not included.

**Example**

A simple example for a transposition cipher is **columnar transposition cipher** where each character in the plain text is written horizontally with specified alphabet width. The cipher is written vertically, which creates an entirely different cipher text.

Consider the plain text **hello world**, and let us apply the simple columnar transposition technique as shown below

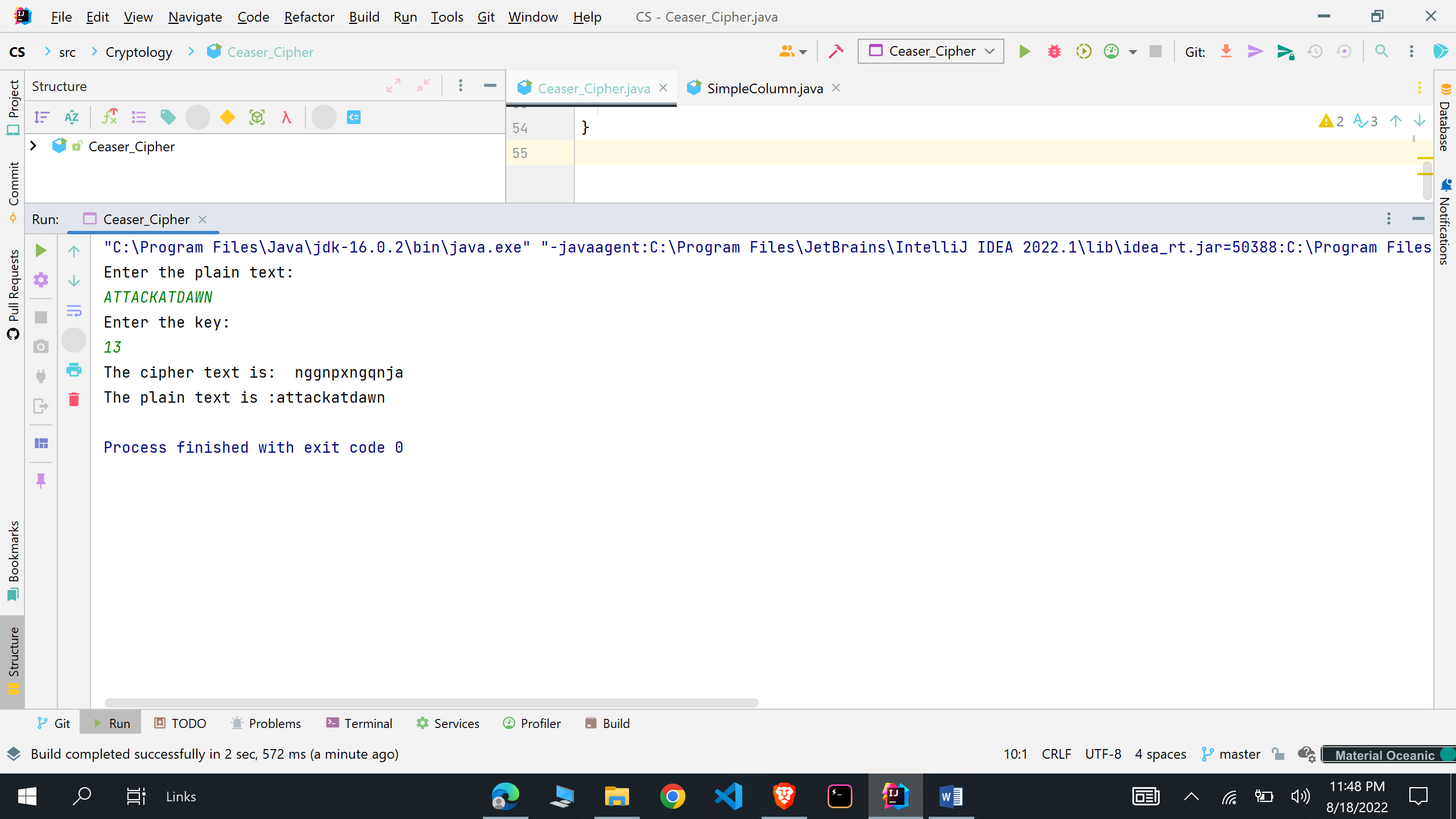


The plain text characters are placed horizontally and the cipher text is created with vertical format as **: holewdlo lr.** Now, the receiver has to use the same table to decrypt the cipher text to plain text

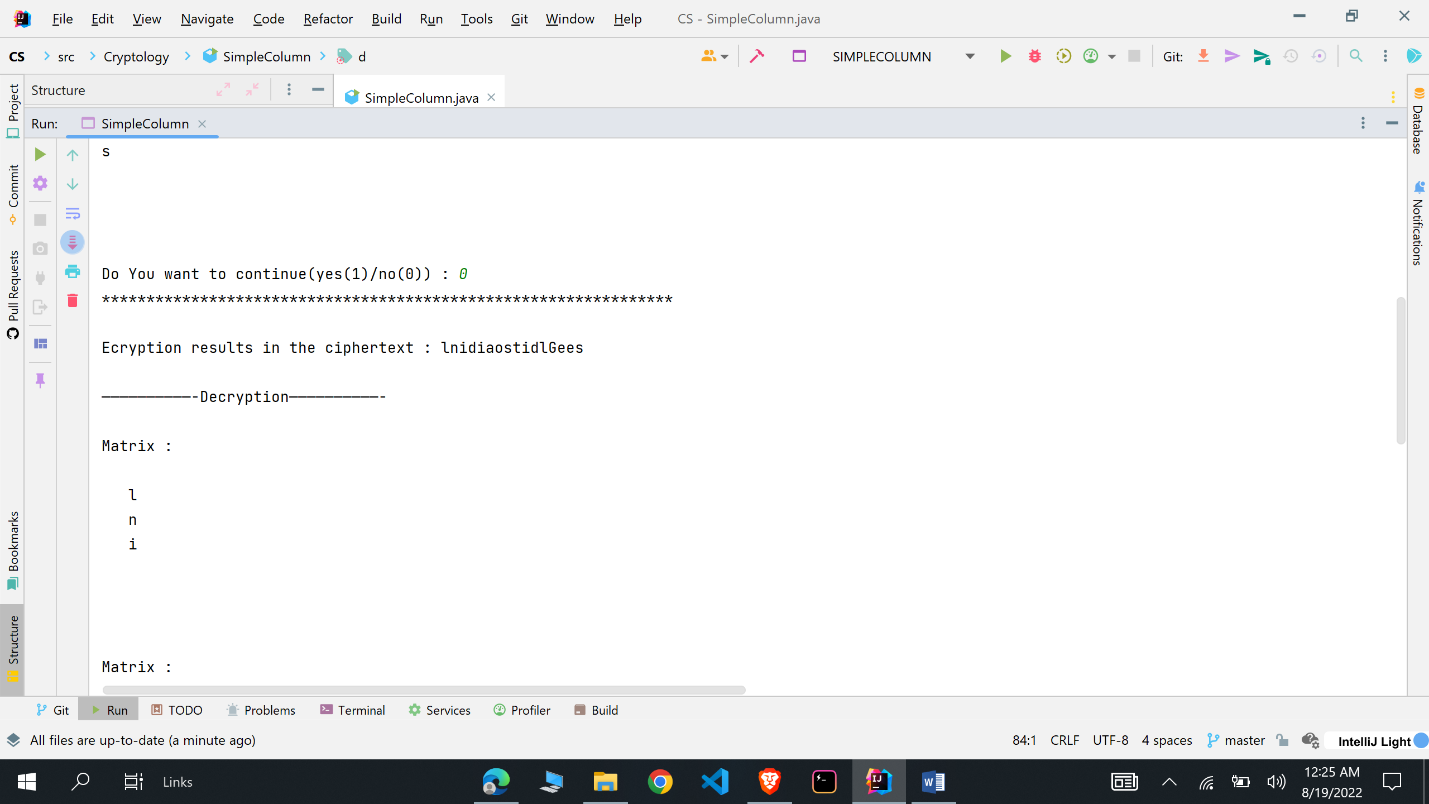
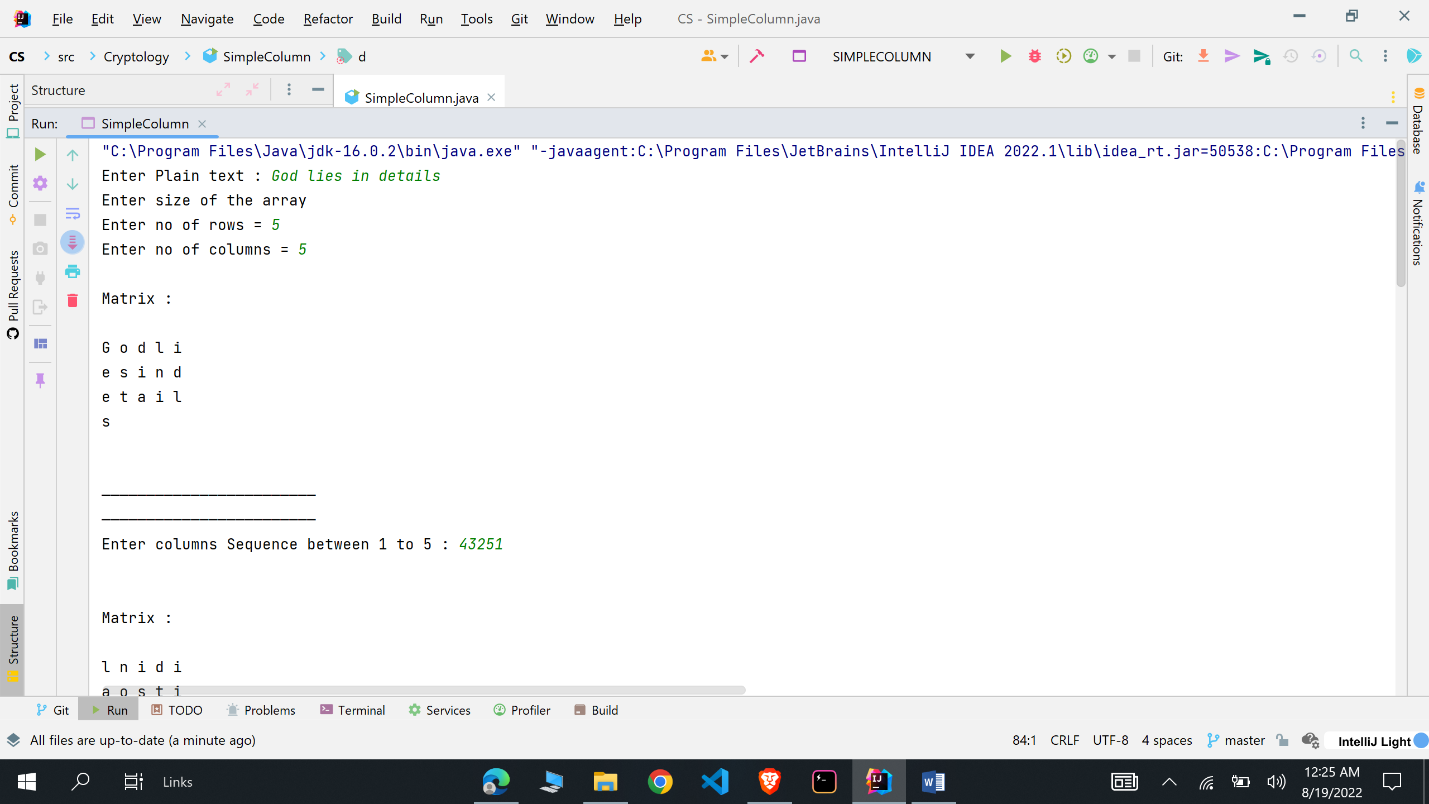
**Program: Ceaser Cipher**

**package Cryptology;  
import java.util.Scanner;**

**public class Ceaser\_Cipher {  
 static String *alphabets* = "abcdefghijklmnopqrstuvwxyz";  
 public static String encoding(String pText , int key){  
 *//converts the user i/p to lower case* String plainText = pText.toLowerCase();  
 String cipherText = "";  
  
 for (int i =0;i<plainText.length();i++){  
 */\*eg : PT= hello , key = 3 here  
 \* plainText.charAt(i) = while i = 0 -> h  
 \* alphabets.indexOf(h) gives 7 therefore charIndex = 7 \*/* int charIndex = *alphabets*.indexOf(plainText.charAt(i));  
 *// newIndex = (7+3)%26 = 10%26 = 0* int newIndex = (charIndex+key)%26;  
 char cpt = *alphabets*.charAt(newIndex);  
 cipherText +=cpt;  
 }  
 return cipherText;  
 }  
  
 public static String decoding (String cipherText , int key){  
 String ptext ="";  
 for (int i=0;i<cipherText.length();i++) {  
 int charIndex = *alphabets*.indexOf(cipherText.charAt(i));  
 int newIndex = (charIndex-key)%26;  
 if (newIndex <0){  
 newIndex = *alphabets*.length()+newIndex;  
 }  
 char plt = *alphabets*.charAt(newIndex);  
 ptext += plt;  
 }  
 return ptext;  
 }  
 public static void main(String[] args) {  
  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("Enter the plain text: ");  
 String pText = sc.nextLine();  
 System.*out*.println("Enter the key: ");  
 int key = sc.nextInt();  
 String cipherText = *encoding* (pText,key) ;  
 System.*out*.println("The cipher text is: "+cipherText);  
 System.*out*.println("The plain text is :"+*decoding*(cipherText,key)); }}**



**package Cryptology;  
import java.io.\*;  
  
public class SimpleColumn  
{  
 static String *s1*,*st*,*d*;  
 static StringBuffer *s*;  
 static int *m*,*n*,*c*,*choice*,*p*,*q*,*k*;  
 static int *z*[]=new int[10];  
 static char *a*[][];  
  
 public static void dis()  
 {  
 System.*out*.println();  
 System.*out*.println("Matrix :");  
 System.*out*.println();  
  
 for(int i=0;i<*m*;i++)  
 {  
 for(int j=0;j<*n*;j++)  
 {  
 if(*a*[i][j]!='$')  
 System.*out*.print(*a*[i][j]+" ");  
else  
 System.*out*.print(" ");  
 }  
 System.*out*.println();  
 }  
 System.*out*.println();  
 }  
  
 public static void enc(DataInputStream dis)throws Exception  
 {  
 while(true)  
 {  
 *c*=0;  
 *s1*="";  
 System.*out*.println("————————————————————————");  
 System.*out*.print("Enter columns Sequence between 1 to "+*n*+" : ");  
 *st*=dis.readLine();  
 *d*=*st*+*d*;  
 System.*out*.println();  
  
 for(int i=0;i<*n*;i++)  
 {  
 *c*=(int)*st*.charAt(i)-49;  
 for(int j=0;j<*m*;j++)  
 {  
 if(*a*[j][*c*]!='$')  
 *s1*=*s1*+*a*[j][*c*];  
 }  
 }  
 *s1*.trim();  
 *c*=0;  
 for(int i=0;i<*m*;i++)  
 for(int j=0;j<*n*;j++)  
 if(*c*<*s1*.length())  
 *a*[i][j]=*s1*.charAt(*c*++);  
 else  
 *a*[i][j]='$';  
 *dis*();  
  
 System.*out*.println();  
 System.*out*.println();  
 System.*out*.print("Do You want to continue(yes(1)/no(0)) : ");  
 *choice*=Integer.*parseInt*(dis.readLine());  
 if(*choice*==0)  
 {  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
 System.*out*.println();  
 System.*out*.println("Ecryption results in the ciphertext : "+*s1*);  
 System.*out*.println();  
  
*//System.exit(0);* return;  
 }  
  
 }  
 }  
  
 public static void dec()  
 {  
 *k*=0;  
 *p*=*s1*.length()/*n*;  
 *q*=*s1*.length()%*n*;  
*//System.out.println("p = "+p+” q = "+q+” d = "+d);* for(int i=0;i<*m*;i++)  
 for(int j=0;j<*n*;j++)  
 *a*[i][j]='$';  
  
 for(int i=0;i<*d*.length();i++)  
 {  
 *c*=(int)*d*.charAt(i)-49;  
*//System.out.println("c = "+c);* if(*c*>=*q*)  
 {  
 for(int j=0;j<*p*;j++)  
 {  
 *a*[j][*c*]=*s1*.charAt(*k*++);  
 }  
 }  
 else  
 {  
 for(int j=0;j<*p*+1;j++)  
 {  
 *a*[j][*c*]=*s1*.charAt(*k*++);  
 }  
 }  
 *dis*();  
 if(*k*==*s1*.length()) {  
 *s1*="";  
 *k*=0;  
 for(int x=0;x<*m*;x++)  
 for(int j=0;j<*n*;j++)  
 if(*a*[x][j]!='$')  
 {  
 *s1*=*s1*+*a*[x][j];  
 *a*[x][j]='$';  
 }  
  
 }  
 }  
 System.*out*.println("Decryption results in the plaintext : "+*s1*);  
 }  
  
 public static void main(String[] args)  
 {  
 try  
 {  
  
 DataInputStream dis=new DataInputStream(System.*in*);  
 System.*out*.print("Enter Plain text : ");  
 *s1*=dis.readLine();  
 *s*=new StringBuffer(*s1*);  
*//REMOVING WIDE-SPACES* for(int i=0;i<*s*.length();i++)  
 if(*s*.charAt(i)==' ')  
 *s*.deleteCharAt(i);  
  
 *s1*=new String(*s*);  
 *d*="";  
 System.*out*.println("Enter size of the array ");  
 System.*out*.print("Enter no of rows = ");  
 *m*=Integer.*parseInt*(dis.readLine());  
 System.*out*.print("Enter no of columns = ");  
 *n*=Integer.*parseInt*(dis.readLine());  
 *a*=new char[*m*][*n*];  
 *c*=0;  
 *//ENTERING IN THE ARRAY* for(int i=0;i<*m*;i++)  
 for(int j=0;j<*n*;j++)  
 if(*c*<*s1*.length())  
 *a*[i][j]=*s1*.charAt(*c*++);  
 else  
 *a*[i][j]='$';  
  
 *dis*();  
 System.*out*.println("————————————————————————");  
 *enc*(dis);  
 System.*out*.println("——————————-Decryption——————————-");  
 *dec*();  
 }  
 catch(Exception e)  
 {} }  
}**

****