**NAME:- G.V.THARUN KUMAR REGD NO:- 19MID0012**

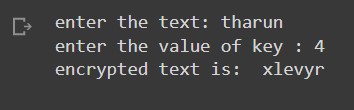
**Applied Cryptography and Network Security**

# CEASER CIPHER:-

CODE:-

## ENCRYPTION:-

n=int(input("\*\*\*\*for encryption enter 1 for decryption enter 2\*\*\*\*")) alph = 'abcdefghijklmnopqrstuvwxyz' if(n==1):

 text1=input("enter the text: ") text=text1.lower() k=int(input("enter the value of key : "))

ctext="" for e in text:

index = alph.find(e) ctext=ctext+alph[(k+index)%27]

print("encrypted text is: ",ctext)

else:

ctext1=input("enter the cypher text: ") ctext2=ctext1.lower() k1=int(input("enter the value of key : ")) dtext="" for e in ctext2:

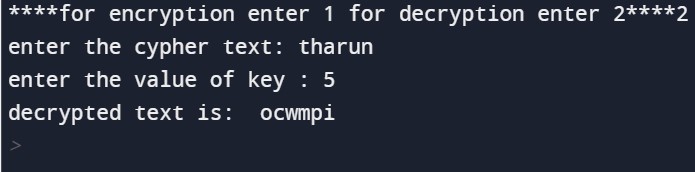
index = alph.find(e) dtext=dtext+alph[(index-k1)%27]

print("decrypted text is: ",dtext)

OUTPUT:-

**DECRYPTION:-**

# OUTPUT:-



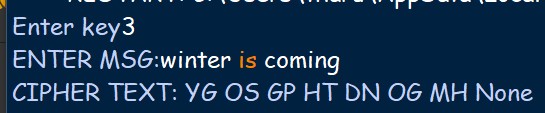
PLAY FAIR CIPHER:-

ENCRYPTION:-

|  |
| --- |
|  |
| key= input ( "Enter key" ) |
| key=key.replace ( " " , "" ) |
| key=key.upper () |
| def matrix ( x , y , initial ) : |
| return [[ initial for i in range ( x )] for j in range (y )] |
|  |
| result= list () |
| for c in key : |
| if c not in result : |
| if c== 'J' : |
| result.append ( 'I' ) |
| else : |
| result.append ( c ) |
| flag= 0 |
| for i in range ( 65 , 91 ): |
| if chr ( i ) not in result : |
| if i== 73 and chr ( 74 ) not in result : |
| result.append ( "I" ) |
| flag= 1 |
| elif flag== 0 and i== 73 or i== 74 : |
| pass |

|  |
| --- |
| else : |
| result.append ( chr ( i )) |
| k= 0 |
| my\_matrix=matrix ( 5 , 5 , 0 ) |
| for i in range ( 0 , 5 ): |
| for j in range ( 0 , 5 ): |
| my\_matrix [ i ][ j ] =result [ k ] |
| k+= 1 |
|  |
| def locindex ( c ) : |
| loc= list () |
| if c== 'J' : |
| c= 'I' |
| for i , j in enumerate ( my\_matrix ): |
| for k , l in enumerate ( j ): |
| if c==l : |
| loc.append ( i ) |
| loc.append ( k ) |
| return loc |
|  |
| def encrypt () : |
| msg= str ( input ( "ENTER MSG:" )) |
| msg=msg.upper () |
| msg=msg.replace ( " " , "" ) |
| i= 0 |
| for s in range ( 0 , len ( msg )+ 1 , 2 ): |
| if s< len ( msg ) -1 : |
| if msg [ s ] ==msg [ s+ 1 ]: |
| msg=msg [: s+ 1 ] + 'X' +msg [ s+ 1 :] |
| if len ( msg ) % 2 != 0 : |
| msg=msg [:] + 'X' |
| print ( "CIPHER TEXT:" , end= ' ' ) |
| while i< len ( msg ): |
| loc= list () |
| loc=locindex ( msg [ i ]) |
| loc1= list () |
| loc1=locindex ( msg [ i+ 1 ]) |
| if loc [ 1 ] ==loc1 [ 1 ]: |
|  |
| print ( "{}{}" . format ( my\_matrix [( loc [ 0 ] + 1 ) % 5 ][ loc [ 1 ]], my\_matrix [( loc1 [ 0 ] + 1 ) % |
| 5 ][ loc1 [ 1 ]]), end= ' ' ) |
| elif loc [ 0 ] ==loc1 [ 0 ]: |
|  |
| print ( "{}{}" . format ( my\_matrix [ loc [ 0 ]][( loc [ 1 ] + 1 ) % 5 ], my\_matrix [ loc1 [0 ]][( lo |
| c1 [ 1 ] + 1 ) % 5 ]), end= ' ' ) |
| else : |
|  |
| print ( "{}{}" . format ( my\_matrix [ loc [ 0 ]][ loc1 [ 1 ]], my\_matrix [ loc1 [ 0 ]][ loc [ 1 ]]) |
| , end= ' ' ) |
| i=i+ 2 |
|  |
| print ( encrypt ()) |
|  |

OUTPUT:-



**HILL CIPHER**

n=int(input("enter the size of key:  "))

keyMatrix = [[0] \* n for i in range(n)]

# Generate vector for the message

messageVector = [[0] for i in range(n)]

# Generate vector for the cipher

cipherMatrix = [[0] for i in range(n)]

# Following function generates the

# key matrix for the key string

def getKeyMatrix(key):

  k = 0

  for i in range(n):

    for j in range(n):

      keyMatrix[i][j] = ord(key[k]) % 65

      k += 1

# Following function encrypts the message

def encrypt(messageVector):

  for i in range(n):

    for j in range(1):

      cipherMatrix[i][j] = 0

      for x in range(n):

        cipherMatrix[i][j] += (keyMatrix[i][x] \*

                  messageVector[x][j])

      cipherMatrix[i][j] = cipherMatrix[i][j] % 26

def HillCipher(message, key):

  # Get key matrix from the key string

  getKeyMatrix(key)

  # Generate vector for the message

  for i in range(n):

    messageVector[i][0] = ord(message[i]) % 65

  # Following function generates

  # the encrypted vector

  encrypt(messageVector)

  # Generate the encrypted text

  # from the encrypted vector

  CipherText = []

  for i in range(n):

    CipherText.append(chr(cipherMatrix[i][0] + 65))

  # Finally print the ciphertext

  print("Ciphertext: ", "".join(CipherText))

# Driver Code

def main():

  # Get the message to

  # be encrypted

  message = input("enter the plain text:  ")

  # Get the key

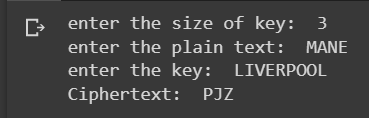
  key = input("enter the key:  ")

  HillCipher(message, key)

if \_\_name\_\_ == "\_\_main\_\_":

  main()

**OUTPUT:-**

****