CEN-593
COMPUTER NETWORK LAB FILE

BTECH COMPUTER ENGINEERING
VTH SEMESTER

SUBMITTED BY:
NABEEL MOHAMMAD RIZWAN

SUBMITTED TO:
PROFESSOR M AMJAD
MR. HANNAN MANSOOR

Department of Computer Engineering,
Faculty of Engineering & Technology,
Jamia Millia Islamia, New Delhi
2022

INDEX

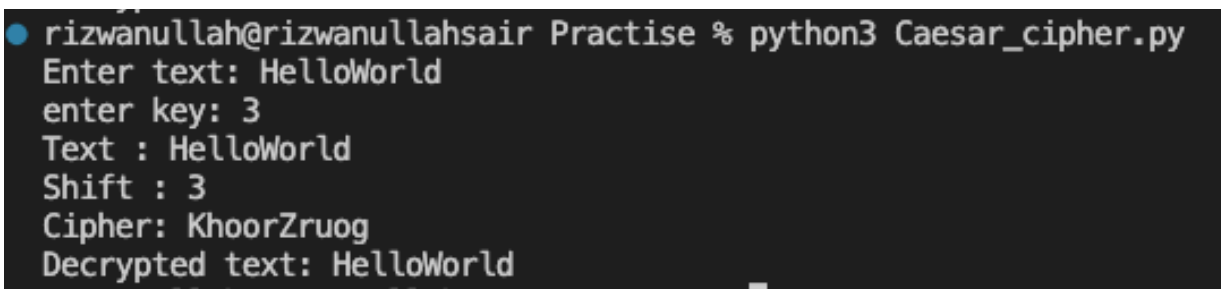
PROGRAM 1:

CAESAR CIPHER
PROGRAMMING LANGUAGE USED : PYTHON

INPUT:

```
def encrypt(text,s):
	result = ""

	for i in range(len(text)):
		char = text[i]

		if (char.isupper()):
			result += chr((ord(char) + s-65) % 26 + 65)

		else:
			result += chr((ord(char) + s - 97) % 26 + 97)

	return result

def decrypt(text,s):
	result = ""

	for i in range(len(text)):
		char = text[i]

		if (char.isupper()):
			result += chr((ord(char) - s - 65) % 26 + 65)

		else:
			result += chr((ord(char) - s - 97) % 26 + 97)

	return result

if __name__ == '__main__':
   text = input("Enter text: ")
   s = input("enter key: ")
   s=int(s)
   print ("Text : " + text)
   print ("Shift : " + str(s))
   print ("Cipher: " + encrypt(text,s))
   text=encrypt(text,s)
   print("Decrypted text: "+ decrypt(text,s))
```

OUTPUT:



PROGRAM 2:

TRANSPOSITION CIPHER
PROGRAMMING LANGUAGE USED:- PYTHON

Input:-

```
import math

key = input("Enter key:-> ")
key=str(key)

def encryptMessage(msg):
	cipher = ""

	k_indx = 0

	msg_len = float(len(msg))
	msg_lst = list(msg)
	key_lst = sorted(list(key))

	col = len(key)

	row = int(math.ceil(msg_len / col))

	fill_null = int((row * col) - msg_len)
	msg_lst.extend('_' * fill_null)

	matrix = [msg_lst[i: i + col]
			for i in range(0, len(msg_lst), col)]

	for _ in range(col):
		curr_idx = key.index(key_lst[k_indx])
		cipher += ''.join([row[curr_idx]
							for row in matrix])
		k_indx += 1

	return cipher

def decryptMessage(cipher):
	msg = ""

	k_indx = 0

	msg_indx = 0
	msg_len = float(len(cipher))
	msg_lst = list(cipher)

	col = len(key)

	row = int(math.ceil(msg_len / col))

	key_lst = sorted(list(key))

	dec_cipher = []
	for _ in range(row):
```

```
                dec_cipher += [[None] * col]

        for _ in range(col):
                curr_idx = key.index(key_lst[k_indx])

                for j in range(row):
                        dec_cipher[j][curr_idx] = msg_lst[msg_indx]
                        msg_indx += 1
                k_indx += 1

        try:
                msg = ''.join(sum(dec_cipher, []))
        except TypeError:
                raise TypeError("This program cannot handle repeating words.")

        null_count = msg.count('_')

        if null_count > 0:
                return msg[: -null_count]

        return msg

msg = input("Enter text:-> ")
msg=str(msg)

cipher = encryptMessage(msg)
print("Encrypted Message: {}".format(cipher))

print("Decryped Message: {}".format(decryptMessage(cipher)))
```

Output:-

```
Enter key:-> HACK
Enter text:-> Sample text
Encrypted Message: aexm tSlept_
Decryped Message: Sample text
```

PROGRAM 3:

BACONIAN CIPHER
PROGRAMMING LANGUAGE USED:- PYTHON

```
lookup = {'A': 'aaaaa', 'B': 'aaaab', 'C': 'aaaba', 'D': 'aaabb', 'E': 'aabaa',
                'F': 'aabab', 'G': 'aabba', 'H': 'aabbb', 'I': 'abaaa', 'J': 'abaab',
                'K': 'ababa', 'L': 'ababb', 'M': 'abbaa', 'N': 'abbab', 'O': 'abbba',
                'P': 'abbbb', 'Q': 'baaaa', 'R': 'baaab', 'S': 'baaba', 'T': 'baabb',
                'U': 'babaa', 'V': 'babab', 'W': 'babba', 'X': 'babbb', 'Y': 'bbaaa', 'Z': 'bbaab'}

def encrypt(message):
        cipher = ''
```

```
        for letter in message:
                if(letter != ' '):

                        cipher += lookup[letter]
                else:
                        cipher += ' '
        return cipher

def decrypt(message):
        decipher = ''
        i = 0
        while True:
                if(i < len(message)-4):
                        substr = message[i:i + 5]
                        if(substr[0] != ' '):

                                decipher += list(lookup.keys())[list(lookup.values()).index(substr)]
                                i += 5

                        else:
                                decipher += ' '
                                i += 1
                else:
                        break

        return decipher

def main():
        message = input("Enter text: ")
        result = encrypt(message.upper())
        print(result)

        message = result
        new = decrypt(message.lower())
        print(new)

if __name__ == '__main__':
        main()
```

```
rizwanullah@rizwanullahsair Practise % python3 Baconian.py
Enter text: hello world
aabbbaabaaababbababbabbba babbaabbbabaaababababbaaabb
HELLO WORLD
```

PROGRAM 4:

SERVER-CLIENT PROGRAM
PROGRAMMING LANGUAGE USED:- PYTHON

Server Side

```
import socket

def socket_program():
   host=socket.gethostname()
```

```
    port=8000
    socket_server=socket.socket()
    socket_server.bind((host,port))
    socket_server.listen(2)
    conn,add=socket_server.accept()
    print("ADDRESS: "+str(add))

    while True:
        data=conn.recv(1024).decode()
        print(data)
        data=input("->")
        conn.send(data.encode())
    conn.close()

if __name__=="__main__":
    socket_program()
```

Client Side:-

```
import socket

def socket_program():
    host=socket.gethostname()
    port=8000
    client_socket=socket.socket()
    client_socket.connect((host,port))

    message=input("->")
    while message != 'bye':
        client_socket.send(message.encode())
        data=client_socket.recv(1024).decode()
        print(data)
        message=input('->')
    client_socket.close()

if __name__=="__main__":
    socket_program()
```

OUTPUT:-

Server

```
rizwanullah@rizwanullahsair Practise % python3 server.py
ADDRESS: ('192.168.1.9', 51138)
hello
->hi
client sending msg here
->server sending msg here
```

Client

```
rizwanullah@rizwanullahsair Practise % python3 client.py
->hello
hi
->client sending msg here
server sending msg here
->bye
rizwanullah@rizwanullahsair Practise %
```

PROGRAM 5:-

SERVER CLIENT USING SUBSTITUTION CIPHER
PROGRAMMING LANGUAGE USED:- PYTHON

Input:-

Server:-

```
import socket

def encrypt(text,s):
  result = ""

  for i in range(len(text)):
        char = text[i]

        if (char.isupper()):
              result += chr((ord(char) + s-65) % 26 + 65)

        else:
              result += chr((ord(char) + s - 97) % 26 + 97)

  return result

def socket_program():
   host=socket.gethostname()
   port=5000
   socket_server=socket.socket()
   socket_server.bind((host,port))
   socket_server.listen(2)
   conn,add=socket_server.accept()
   print("ADDRESS: "+str(add))

   while True:
      data=conn.recv(1024).decode()
      print(data)
      data=input("->")
      s = input("enter key: ")
      s=int(s)
      print ("Text : " + data)
      print ("Shift : " + str(s))
      print ("Cipher: " + encrypt(data,s))

      a=encrypt(data,s)
      s=str(s)
      a=str(a)
      data=a+s
      conn.send(data.encode())

   conn.close()

if __name__=="__main__":
   socket_program()
```

Client:-

```
import socket

def decrypt(text,s):
  result = ""

  for i in range(len(text)):
            char = text[i]

            if (char.isupper()):
                      result += chr((ord(char) - s - 65) % 26 + 65)

            else:
                      result += chr((ord(char) - s - 97) % 26 + 97)

  return result

def socket_program():
   host=socket.gethostname()
   port=5000
   client_socket=socket.socket()
   client_socket.connect((host,port))

   message=input("->")
   while message != 'bye':
      client_socket.send(message.encode())
      data=client_socket.recv(1024).decode()
      key=input("Enter key to receive decrypted message from server:-> ")
      key=int(key)
      s=data[-1]
      s=int(s)
      text=data[:-1]
      text=str(text)
      print(s)
      if(key==s):
         print("Decrypted text: "+ decrypt(text,s))
         message=input('->')
      else:
         print("wrong key entered, terminating program")
         break
   client_socket.close()

if __name__=="__main__":
   socket_program()
```

Output:-

Server side

```
rizwanullah@rizwanullahsair Substitution_cipher.py % python3 serv
er.py
ADDRESS: ('192.168.1.6', 58211)
hi
->hello
enter key: 3
Text : hello
Shift : 3
Cipher: khoor
```

Client Side

```
rizwanullah@rizwanullahsair Substitution_cipher.py % python3 cli
ent.py
->hi
Enter key to receive decrypted message from server:-> 3
3
Decrypted text: hello
->bye
rizwanullah@rizwanullahsair Substitution_cipher.py %
```

PROGRAM 6:

SERVER-CLIENT MULTIPLE CLIENT SOCKET PROGRAM
PROGRAMMING LANGUAGE USED:- PYTHON

Server:-

```
import socket
from _thread import *

ServerSocket = socket.socket()
host = '127.0.0.1'
port = 1233
ThreadCount = 0
try:
    ServerSocket.bind((host, port))
except socket.error as e:
    print(str(e))

print('Waiting for a Connection..')
ServerSocket.listen(5)

def threaded_client(connection):
    connection.send(str.encode('Welcome to the Server'))
    while True:
        data = connection.recv(2048)
        reply = 'Server Says: ' + data.decode('utf-8')
        if not data:
            break
        connection.sendall(str.encode(reply))
    connection.close()

while True:
    Client, address = ServerSocket.accept()
    print('Connected to: ' + address[0] + ':' + str(address[1]))
    start_new_thread(threaded_client, (Client, ))
    ThreadCount += 1
    print('Clients connected:-> ' + str(ThreadCount))
ServerSocket.close()
```

Client:-

```
import socket
from _thread import *
```

```
ClientSocket = socket.socket()
host = '127.0.0.1'
port = 1233

print('Waiting for connection')
try:
   ClientSocket.connect((host, port))
except socket.error as e:
   print(str(e))

Response = ClientSocket.recv(1024)

while True:
   Input = input('Say Something: ')
   ClientSocket.send(str.encode(Input))
   Response2 = ClientSocket.recv(1024)
   print(Response2.decode('utf-8'))


ClientSocket.close()
```

Output:-

Server

```
ltiple_server_client % python3
 servermultiple.py
Waiting for a Connection..
Connected to: 127.0.0.1:59293
Clients connected:-> 1
Connected to: 127.0.0.1:59303
Clients connected:-> 2
Connected to: 127.0.0.1:59329
Clients connected:-> 3
```

Client 1

```
rizwanullah@rizwanullahsair Mu
ltiple_server_client % python3
 clientmultiple.py
Waiting for connection
Say Something: hi
Server Says: hi
Say Something: 1st client
Server Says: 1st client
Say Something:
```

Client 2

```
rizwanullah@rizwanullahsair Mu
ltiple_server_client % python3
 clientmultiple.py
Waiting for connection
Say Something: 3rd client
Server Says: 3rd client
Say Something: computer networ
k lab
Server Says: computerk lab
Say Something: 2nd client
Server Says: 2nd client
Say Something:
```

Client 3

```
rizwanullah@rizwanullahsair M
ultiple_server_client % pytho
n3 clientmultiple.py
Waiting for connection
Say Something: hello
Server Says: hello
Say Something: 3rd client
Server Says: 3rd client
Say Something:
```

PROGRAM 7:-

CHECK THE DATATYPE OF INPUT FROM THE CLIENT ON THE SERVER
PROGRAMMING LANGUAGE USED:- PYTHON

Input:-

Server:-

```
import socket

def server_program():
   host = socket.gethostname()
   port = 8000

   server_socket = socket.socket()
   server_socket.bind((host, port))

   server_socket.listen(2)
   conn, address = server_socket.accept()
   print("Connection from: " + str(address))
   while True:
```

```
        data = conn.recv(1024).decode()
        if not data:
            break

        print("from connected user: " + str(data))

        sp="@#$%^&*()!±?/.,<>|;:]}[{"
        no="1234567890"
        space=" "

        if any(i in sp for i in data):
            print("Special Characters present in text from client")

        elif any (i in no for i in data):
            print("Number present in text send by client")

        elif any (i in space for i in data):
            print("space present in text send by client")

        else:
            print("No special Characters or numbers or space present in text from client")

        data = input(' -> ')
        conn.send(data.encode())

    conn.close()

if __name__ == '__main__':
    server_program()
```

Client:-

```
import socket

def client_program():
    host = socket.gethostname()
    port = 8000

    client_socket = socket.socket()
    client_socket.connect((host, port))

    message = input(" -> ")

    while message.lower().strip() != 'bye':
        client_socket.send(message.encode())
        data = client_socket.recv(1024).decode()

        print('Received from server: ' + data)

        message = input(" -> ")

    client_socket.close()

if __name__ == '__main__':
    client_program()
```

Output:-

Server Side

```
rizwanullah@rizwanullahsair Server_Client_program % python3 serve
r.py
Connection from: ('192.168.1.6', 52505)
from connected user: hello
No special Characters or numbers or space present in text from cl
ient
 -> hi
from connected user: @#
Special Characters present in text from client
 -> send msg for checking numbers
from connected user: 123
Number present in text send by client
 -> checking for spaces
from connected user: hello world
space present in text send by client
 -> bye
```

Client side

```
rizwanullah@rizwanullahsair Server_Client_program % python3 clie
nt.py
 -> hello
Received from server: hi
 -> @#
Received from server: send msg for checking numbers
 -> 123
Received from server: checking for spaces
 -> hello world
Received from server: bye
 -> bye
rizwanullah@rizwanullahsair Server_Client_program %
```

PROGRAM 8:-

RAIL FENCING CIPHER USING SERVER-CLIENT PROGRAMMING LANGUAGE USED:- PYTHON

Input:-

Server:-

```
import socket

def decryptRailFence(data, key):

    rail = [['\n' for i in range(len(data))]
                for j in range(key)]
    dir_down = None
    row, col = 0, 0

    for i in range(len(data)):
        if row == 0:
            dir_down = True
        if row == key - 1:
            dir_down = False

        rail[row][col] = '*'
        col += 1

        if dir_down:
            row += 1
        else:
            row -= 1

    index = 0
    for i in range(key):
        for j in range(len(data)):
            if ((rail[i][j] == '*') and
              (index < len(data))):
                rail[i][j] = data[index]
                index += 1

    result = []
```

```
    row, col = 0, 0
    for i in range(len(data)):

        if row == 0:
            dir_down = True
        if row == key-1:
            dir_down = False

        if (rail[row][col] != '*'):
            result.append(rail[row][col])
            col += 1

        if dir_down:
            row += 1
        else:
            row -= 1
    return("".join(result))

def server_program():
    host = socket.gethostname()
    port = 5000

    server_socket = socket.socket()
    server_socket.bind((host, port))

    server_socket.listen(2)
    conn, address = server_socket.accept()
    print("Connection from: " + str(address))
    while True:
        data = conn.recv(1024).decode()
        if not data:
            break

        print("Enter key: ")
        authenticate=input()
        if(authenticate == data[len(data)-1]):
            print("from connected user: " + str(data))
            key=int(data[len(data)-1])
            data=data.rstrip(data[-1])
            print(decryptRailFence(data, key))

        else:
            print("Wrong key, Cannot access message")

        data = input(' -> ')
        conn.send(data.encode())
    conn.close()

if __name__ == '__main__':
   server_program()
```

Client:-

```
import socket

def encryptRailFence(message, key):
```

```
    key=int(key)

    rail = [['\n' for i in range(len(message))]
                for j in range(key)]

    dir_down = False
    row, col = 0, 0

    for i in range(len(message)):

        if (row == 0) or (row == key - 1):
            dir_down = not dir_down

        rail[row][col] = message[i]
        col += 1

        if dir_down:
            row += 1
        else:
            row -= 1
    result = []
    for i in range(key):
        for j in range(len(message)):
            if rail[i][j] != '\n':
                result.append(rail[i][j])
    return("" . join(result))

def client_program():
    host = socket.gethostname()
    port = 5000

    client_socket = socket.socket()

    client_socket.connect((host, port))

    message = input(" -> ")

    while message.lower().strip() != 'bye':
        print("enter key: ")
        key=input()
        print(encryptRailFence(message, key))
        encrypted=encryptRailFence(message, key)
        client_socket.send(encrypted.encode())
        client_socket.send(key.encode())
        data = client_socket.recv(1024).decode()
        print('Received from server: ' + data)

        message = input(" -> ")
    client_socket.close()

if __name__ == '__main__':
    client_program()
```

Output:

Server Side

```
rizwanullah@rizwanullahsair Encryption_server_client % python3 se
rver_en.py
Connection from: ('192.168.1.6', 51996)
Enter key:
3
from connected user: hoell3
hello
 -> received encrypted message
Enter key:
4
from connected user: okay4
okay
 -> bye
rizwanullah@rizwanullahsair Encryption_server_client %
```

Client side

```
rizwanullah@rizwanullahsair Encryption_server_client % python3 c
lient_en.py
 -> hello
enter key:
3
hoell
Received from server: received encrypted message
 -> okay
enter key:
4
okay
Received from server: bye
 -> bye
rizwanullah@rizwanullahsair Encryption_server_client %
```

PROGRAM 9:-

SERVER-CLIENT WITH VIGENERE CIPHER
PROGRAMMING LANGUAGE USED:- PYTHON

Input:-

Server:-

```
import socket

def generateKey(string, key):
        key = list(key)
        if len(string) == len(key):
                return(key)
        else:
                for i in range(len(string) -
                                                len(key)):
                        key.append(key[i % len(key)])
        return("" . join(key))

def originalText(cipher_text, key):
        orig_text = []
        for i in range(len(cipher_text)):
                x = (ord(cipher_text[i]) -
                        ord(key[i]) + 26) % 26
                x += ord('A')
                orig_text.append(chr(x))
        return("" . join(orig_text))

def server_program():
    host = socket.gethostname()
    port = 5000

    server_socket = socket.socket()
    server_socket.bind((host, port))

    server_socket.listen(2)
    conn, address = server_socket.accept()
    print("Connection from: " + str(address))
    while True:
```

```
        data = conn.recv(1024).decode()
        if not data:
           break
        print("\n")
        print('20BCS087 NABEEL MOHAMMAD RIZWAN')
        print("\n")
        print("Encrypted message: ")
        print(data)
        print("\n")
        keyword=input("Enter keyword set on client side to decrypt message: ")
        key = generateKey(data, keyword)
        print("Decrypted Text :",originalText(data, key))
        Decrypted=originalText(data, key)
        print("from connected user: " + str(Decrypted))

        data = input(' -> ')
        conn.send(data.encode())

    conn.close()

if __name__ == '__main__':
   server_program()
```

Client:-

```
import socket

def generateKey(string, key):
        key = list(key)
        if len(string) == len(key):
                return(key)
        else:
                for i in range(len(string) -
                                                len(key)):
                        key.append(key[i % len(key)])
        return("" . join(key))

def cipherText(string, key):
        cipher_text = []
        for i in range(len(string)):

                x = (ord(string[i]) +
                        ord(key[i])) % 26
                x += ord('A')
                cipher_text.append(chr(x))
        return("" . join(cipher_text))

def originalText(cipher_text, key):
        orig_text = []
        for i in range(len(cipher_text)):
                x = (ord(cipher_text[i]) -
                        ord(key[i]) + 26) % 26
                x += ord('A')
                orig_text.append(chr(x))
        return("" . join(orig_text))

def client_program():
```

```
    host = socket.gethostname()
    port = 5000

    client_socket = socket.socket()
    client_socket.connect((host, port))

    message = input(" -> ")

    keyword=input("Enter keyword ")
    key = generateKey(message, keyword)
    cipher_text = cipherText(message,key)
    print("Ciphertext :", cipher_text)
    print("Original/Decrypted Text :",originalText(cipher_text, key))

    while message.lower().strip() != 'bye':
        client_socket.send(cipher_text.encode())
        data = client_socket.recv(1024).decode()
        print("\n")
        print('20BCS087 NABEEL MOHAMMAD RIZWAN')
        print('Received from server: ' + data)
        print('characters received from server: ')
        message = input(" -> ")

        keyword=input("Enter keyword ")
        key = generateKey(message, keyword)
        cipher_text = cipherText(message,key)
        print("Ciphertext :", cipher_text)
        print("Original/Decrypted Text :",originalText(cipher_text, key))
    client_socket.close()

if __name__ == '__main__':
    client_program()
```

Output:-

Server Side

```
rizwanullah@rizwanullahsair Cipher_server_client % python3 server
.py
Connection from: ('192.168.1.6', 53051)


20BCS087 NABEEL MOHAMMAD RIZWAN


Encrypted message:
ZEXAZAGRXS


Enter keyword set on client side to decrypt message: SAMPLE
Decrypted Text : HELLOWORLD
from connected user: HELLOWORLD
 -> hi
```

Client Side

```
t.py
 -> HELLOWORLD
Enter keyword SAMPLE
Ciphertext : ZEXAZAGRXS
Original/Decrypted Text : HELLOWORLD


20BCS087 NABEEL MOHAMMAD RIZWAN
Received from server: hi
characters received from server:
 -> bye
Enter keyword bye
Ciphertext : OIU
Original/Decrypted Text : HEK
rizwanullah@rizwanullahsair Cipher_server_client %
```

PROGRAM 10:-

PLAYFAIR CIPHER
PROGRAMMING LANGUAGE USED:- PYTHON

Input:-

```
def toLowerCase(text):
	return text.lower()

def removeSpaces(text):
	newText = ""
	for i in text:
		if i == " ":
			continue
		else:
			newText = newText + i
	return newText

def Diagraph(text):
	Diagraph = []
	group = 0
	for i in range(2, len(text), 2):
		Diagraph.append(text[group:i])

		group = i
	Diagraph.append(text[group:])
	return Diagraph

def FillerLetter(text):
	k = len(text)
	if k % 2 == 0:
		for i in range(0, k, 2):
			if text[i] == text[i+1]:
				new_word = text[0:i+1] + str('x') + text[i+1:]
				new_word = FillerLetter(new_word)
				break
			else:
				new_word = text
	else:
		for i in range(0, k-1, 2):
			if text[i] == text[i+1]:
				new_word = text[0:i+1] + str('x') + text[i+1:]
				new_word = FillerLetter(new_word)
				break
			else:
				new_word = text
	return new_word

list1 = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k', 'l', 'm',
		'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

def generateKeyTable(word, list1):
	key_letters = []
	for i in word:
		if i not in key_letters:
```

```
                        key_letters.append(i)

    compElements = []
    for i in key_letters:
        if i not in compElements:
            compElements.append(i)
    for i in list1:
        if i not in compElements:
            compElements.append(i)
    matrix = []
    while compElements != []:
        matrix.append(compElements[:5])
        compElements = compElements[5:]
    return matrix

def search(mat, element):
    for i in range(5):
        for j in range(5):
            if(mat[i][j] == element):
                return i, j

def encrypt_RowRule(matr, e1r, e1c, e2r, e2c):
    char1 = ''
    if e1c == 4:
        char1 = matr[e1r][0]
    else:
        char1 = matr[e1r][e1c+1]
    char2 = ''
    if e2c == 4:
        char2 = matr[e2r][0]
    else:
        char2 = matr[e2r][e2c+1]

    return char1, char2

def encrypt_ColumnRule(matr, e1r, e1c, e2r, e2c):
    char1 = ''
    if e1r == 4:
        char1 = matr[0][e1c]
    else:
        char1 = matr[e1r+1][e1c]

    char2 = ''
    if e2r == 4:
        char2 = matr[0][e2c]
    else:
        char2 = matr[e2r+1][e2c]

    return char1, char2

def encrypt_RectangleRule(matr, e1r, e1c, e2r, e2c):
    char1 = ''
    char1 = matr[e1r][e2c]

    char2 = ''
    char2 = matr[e2r][e1c]
```

```
        return char1, char2

def encryptByPlayfairCipher(Matrix, plainList):
        CipherText = []
        for i in range(0, len(plainList)):
                c1 = 0
                c2 = 0
                ele1_x, ele1_y = search(Matrix, plainList[i][0])
                ele2_x, ele2_y = search(Matrix, plainList[i][1])

                if ele1_x == ele2_x:
                        c1, c2 = encrypt_RowRule(Matrix, ele1_x, ele1_y, ele2_x, ele2_y)
                elif ele1_y == ele2_y:
                        c1, c2 = encrypt_ColumnRule(Matrix, ele1_x, ele1_y, ele2_x, ele2_y)
                else:
                        c1, c2 = encrypt_RectangleRule(
                                Matrix, ele1_x, ele1_y, ele2_x, ele2_y)

                cipher = c1 + c2
                CipherText.append(cipher)
        return CipherText
print("\n")
print("20BCS087 NABEEL MOHAMMAD RIZWAN\n")
text_Plain = input("Enter text: ")
text_Plain=str(text_Plain)
text_Plain = removeSpaces(toLowerCase(text_Plain))
PlainTextList = Diagraph(FillerLetter(text_Plain))
if len(PlainTextList[-1]) != 2:
        PlainTextList[-1] = PlainTextList[-1]+'z'

key = input("Enter keyword: ")
key=str(key)
print("Key text:", key)
key = toLowerCase(key)
Matrix = generateKeyTable(key, list1)

print("Plain Text:", text_Plain)
CipherList = encryptByPlayfairCipher(Matrix, PlainTextList)
CipherText = ""
for i in CipherList:
        CipherText += i
print("CipherText:", CipherText)
```

Output:-

```
rizwanullah@rizwanullahsair Lab4 % python3 lab.py


20BCS087 NABEEL MOHAMMAD RIZWAN

Enter text: instruments
Enter keyword: Monarchy
Key text: Monarchy
Plain Text: instruments
CipherText: gatlmzclrqtx
```