OUTPUTS:

client.py:

```
import socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = socket.gethostname()

port = 3000

client_socket.connect((host, port))

p = 53
q = 59
message = input("Enter the message : ")
n = p * q
phiN = (p-1)*(q-1)
e = 3
k = 2
d = ((k*phiN)+1)/e

print("Message is: {0}".format(message))

public_key = str(e)+","+str(n)
print("Public key is: {0}".format(public_key))

private_key = [int(d), n]
print("Private key is: {0}".format(private_key))
print("------------------------------------------------------------")
message_to_send = str(str(message)+","+public_key)
client_socket.send(str(message_to_send).encode("ascii"))

received_data = client_socket.recv(1024).decode("ascii")
print("Received response: {0}".format(received_data))

decrypted_message = pow(int(received_data),int(d))%n
print("Decrypted message: {0}".format(decrypted_message))

client_socket.close()
```

server.py:

```
import socket

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = socket.gethostname()
port = 3000
server_socket.bind((host, port))
server_socket.listen(1)

while True:
	client_socket, addr = server_socket.accept()
	print("Got connection from {0}".format(addr))

	received_data = client_socket.recv(1024).decode("ascii")
	print("Received data from client: {0}".format(received_data))

	received_data = received_data.split(",")
	message = received_data[0]
	public_key = [received_data[1], received_data[2]]
	e = int(public_key[0])
	n = int(public_key[1])

	print("Message is: {0}".format(message))
	print("Public Key is: {0}".format(public_key))

	cypher_text = pow(int(message), e)
	cypher_text = cypher_text%n
	print("Cypher text is: {0}".format(cypher_text))
	client_socket.send(str(cypher_text).encode("ascii"))
	client_socket.close()
```

main.py:

```
p = 53
q = 59
plain_text = "542"

n = p * q
phiN = (p-1)*(q-1)

e = 3
message_raised_to_e = pow(int(plain_text), e)
cypher_text = message_raised_to_e%n

def check_for_prime(num):
	flag = False
	if num==1:
```

```
            return False
        elif num>1:
            for i in range(2, num):
                if(num%i)==0:
                    flag = True
                    break
            if flag:
                return False
            else:
                return True

public_key = [e, n]
print(public_key)

p = 53
q = 59

k = 2
d = ((k*phiN)+1)/e

private_key = [int(d), n]
print(private_key)

m = pow(cypher_text,int(d))%n
print(m)
```