1.1) Write a C Program to implement Shift Cipher.

```
#include <stdio.h>
#include <string.h>

void encrypt(char pt[], int shift) {
   for (int i = 0; i < strlen(pt); i++) {
      if (pt[i] >= 'a' && pt[i] <= 'z') {
         pt[i] = ((pt[i] - 'a' + shift) % 26) + 'a';
      }
      else if (pt[i] >= 'A' && pt[i] <= 'Z') {
         pt[i] = ((pt[i] - 'A' + shift) % 26) + 'A';
      }
   }
}

void decrypt(char pt[], int shift) {
   for (int i = 0; i < strlen(pt); i++) {
      if (pt[i] >= 'a' && pt[i] <= 'z') {
         pt[i] = ((pt[i] - 'a' - shift + 26) % 26) + 'a';
      }
      else if (pt[i] >= 'A' && pt[i] <= 'Z') {
         pt[i] = ((pt[i] - 'A' - shift + 26) % 26) + 'A';
      }
   }
}

int main() {
   char pt[100];
   int shift;

   printf("Enter a plain text:\n");
   gets(pt);

   printf("Enter shift value:\n");
   scanf("%d", &shift);

   encrypt(pt, shift);
   printf("The Encryption is %s\n", pt);

   decrypt(pt, shift);
   printf("The decrypt is %s\n", pt);

   return 0;
}
```

1.2) Write a C Program to implement Mono-Alphabetic Substitution Cipher.

```
#include <stdio.h>
#include <string.h>

void encrypt(char pt[], char key[]) {
   int i;
   for (i = 0; i < strlen(pt); i++) {
      if (pt[i] >= 'a' && pt[i] <= 'z') {
         pt[i] = key[pt[i] - 'a'];
      }
      else if (pt[i] >= 'A' && pt[i] <= 'Z') {
         pt[i] = key[pt[i] - 'A'];
      }
   }
}

void decrypt(char pt[], char key[]) {
   char reverseKey[26];
   int i;
   for (i = 0; i < 26; i++) {
      reverseKey[key[i] - 'A'] = 'A' + i;
   }
   for (i = 0; i < strlen(pt); i++) {
      if (pt[i] >= 'a' && pt[i] <= 'z') {
         pt[i] = reverseKey[pt[i] - 'a'];
      }
      else if (pt[i] >= 'A' && pt[i] <= 'Z') {
         pt[i] = reverseKey[pt[i] - 'A'];
      }
   }
}

int main() {
   char pt[100];
   char key[26] = "QWERTYUIOPLKJHGFDSAZXCVBNM";

   printf("Enter a plainText: ");
   gets(pt);

   encrypt(pt, key);
```

```
    printf("Encrypted message: %s\n", pt);

    decrypt(pt, key);
    printf("Decrypted message: %s\n", pt);

    return 0;
  }
```

2.1) Write a C Program to implement one-time pad cipher.

```
#include <stdio.h>

#include <string.h>


void encrypt(char pt[], char key[], char ct[]) {

  int i;

  for (i = 0; pt[i] != '\0'; i++) {

    ct[i] = ((pt[i] - 'A') + (key[i] - 'A')) % 26 + 'A';

  }

  ct[i] = '\0';

}


void decrypt(char ct[], char key[], char pt[]) {

  int i;

  for (i = 0; ct[i] != '\0'; i++) {

    pt[i] = ((ct[i] - 'A') - (key[i] - 'A') + 26) % 26 + 'A';

  }

  pt[i] = '\0';

}
```

```
int main() {
    char pt[100], key[100], ct[100], dt[100];

    printf("Enter pt (uppercase letters only): ");
    scanf("%s", pt);

    printf("Enter key (same length as pt): ");
    scanf("%s", key);

    if (strlen(pt) != strlen(key)) {
        printf("Error: Key must be the same length as pt.\n");
        return 1;
    }

    encrypt(pt, key, ct);
    printf("Ciphertext: %s\n", ct);

    decrypt(ct, key, dt);
    printf("Decrypted text: %s\n", dt);

    return 0;
}
```

2.2) Write a C Program to implement vernam cipher.

```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>

void encrypt(char *plaintext, char *key, char *ciphertext) {

   int i;

   for (i = 0; i < strlen(plaintext); i++) {

      ciphertext[i] = plaintext[i] ^ key[i];

   }

   ciphertext[i] = '\0'; // Ensure null termination

}

void decrypt(char *ciphertext, char *key, char *plaintext) {

   int i;

   for (i = 0; i < strlen(ciphertext); i++) {

      plaintext[i] = ciphertext[i] ^ key[i];

   }

   plaintext[i] = '\0'; // Ensure null termination

}


int main() {

   char plaintext[100], key[100], ciphertext[100];

   printf("Enter Plaintext: ");

   scanf("%s", plaintext);
```

```
    printf("Enter Key (same length as Plaintext): ");

    scanf("%s", key);

    if (strlen(plaintext) != strlen(key)) {

        printf("Error: Key must be the same length as plaintext.\n");

        return 1;

    }

    encrypt(plaintext, key, ciphertext);

    printf("Ciphertext: ");

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

        printf("%02X", (unsigned char)ciphertext[i]);

    }

    printf("\n");

    decrypt(ciphertext, key, plaintext);

    printf("Decrypted Text: %s\n", plaintext);

    return 0;

}
```

4.1) Write a C Program to implement RSA algorithm.

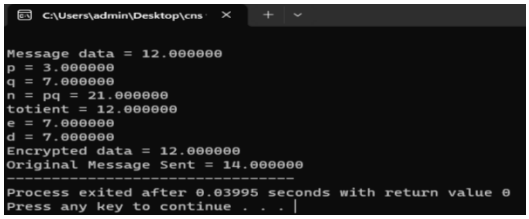
PROGRAM:

```
#include<stdio.h>
#include<math.h>
//to find gcd
int gcd(int a, int b)
{
if (b == 0) return a;
return gcd(b, a % b);
}
int main()
{
//2 random prime numbers
double p = 3;
double q = 7;
```

```
double n=p*q;
int count;
double totient = (p-1)*(q-1);
double e=6;
while(e<totient)
{
count = gcd(e,totient);
if(count==1)
break;
else
e++;
}
double d,i;
for(i=1;i<totient;i++)
{
if(fmod(e*i,totient)==1.0)
{
d=i; break;
}
}

double msg = 12;
double c = pow(msg,e);
double m = pow(c,d);
c=fmod(c,n);
m=fmod(m,n);
printf("\nMessage data = %lf",msg);
printf("\np = %lf",p);
printf("\nq = %lf",q);
printf("\nn = pq = %lf",n);
printf("\ntotient = %lf",totient);
printf("\ne = %lf",e);
printf("\nd = %lf",d);
printf("\nEncrypted data = %lf",c);
printf("\nOriginal Message Sent = %lf",m);
return 0;
}
```

OUTPUT



4.3) Write a C Program to implement Elgamal Cryptographic System.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
long long int mod_exp(long long int base, long long int exp, long long int mod) {
long long int result = 1;
while (exp > 0) {
if (exp % 2 == 1) {
result = (result * base) % mod;
}
exp = exp >> 1;
base = (base * base) % mod;
}
return result;
}
int main() {
long long int p = 11;
long long int e1 = 2;
long long int d = 3;
long long int e2 = mod_exp(e1, d, p);
printf("Public Key: {p = %lld, e1 = %lld, e2 = %lld}\n", p, e1, e2);
printf("Private Key: {d = %lld}\n", d);
long long int M;
printf("Enter message (integer < %lld): ", p);
scanf("%lld", &M);
if (M >= p) {
printf("Error: Message must be smaller than p (%lld)\n", p);
return 1;
}
long long int r = 3; // Random integer (ephemeral key)
long long int c1 = mod_exp(e1, r, p);
long long int c2 = (M * mod_exp(e2, r, p)) % p;
printf("Encrypted Message: {c1 = %lld, c2 = %lld}\n", c1, c2);
long long int M_decrypted = (c2 * mod_exp(c1, p - 1 - d, p)) % p;
printf("Decrypted Message: %lld\n", M_decrypted);
return 0;
```

```
Public Key: {p = 11, e1 = 2, e2 = 8}
Private Key: {d = 3}
Enter message (integer < 11): 9
Encrypted Message: {c1 = 8, c2 = 10}
Decrypted Message: 9

--------------------------------
```

Experiment-11

11) Implement file transfer using Message Queue form of IPC.

PROGRAM:

Sender:-

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define MSG_SIZE 256
#define FILE_PATH "input.txt"
#define QUEUE_KEY 1234
struct msg_buffer {
long msg_type;
char msg_text[MSG_SIZE];
};
int main() {
key_t key = QUEUE_KEY;
int msgid;
struct msg_buffer message;
FILE *file;
msgid = msgget(key, 0666 | IPC_CREAT);
if (msgid == -1) {
perror("msgget");
exit(EXIT_FAILURE);
}
file = fopen(FILE_PATH, "r");
if (file == NULL) {
perror("fopen");
exit(EXIT_FAILURE);
}
message.msg_type = 1;
while (fgets(message.msg_text, MSG_SIZE, file) != NULL) {
if (msgsnd(msgid, &message, sizeof(message.msg_text), 0) == -1)
{ perror("msgsnd");
exit(EXIT_FAILURE);
}}
strcpy(message.msg_text, "EOF");
msgsnd(msgid, &message, sizeof(message.msg_text), 0);
printf("File sent successfully.\n");
fclose(file);
return 0;
}
```

OUTPUT:

```
[22A91A05K1@Linux ~]$ vi cns11sender.c
[22A91A05K1@Linux ~]$ cc -o cns11sender cns11sender.c
[22A91A05K1@Linux ~]$ ./cns11sender
File sent successfully.
[22A91A05K1@Linux ~]$
```

Receiver:-

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define MSG_SIZE 256
#define FILE_PATH "output.txt"
#define QUEUE_KEY 1234
struct msg_buffer {
long msg_type;
char msg_text[MSG_SIZE];
};
int main() {
key_t key = QUEUE_KEY;
int msgid;
struct msg_buffer message;
FILE *file;
msgid = msgget(key, 0666 | IPC_CREAT);
if (msgid == -1) {
perror("msgget");
exit(EXIT_FAILURE);
}
file = fopen(FILE_PATH, "w");
if (file == NULL) {
perror("fopen");
exit(EXIT_FAILURE);
}
while (1) {
if (msgrcv(msgid, &message, sizeof(message.msg_text), 1, 0) == -1)
{ perror("msgrcv");
exit(EXIT_FAILURE);
}
if (strcmp(message.msg_text, "EOF") == 0)
{ break;
}
fprintf(file, "%s", message.msg_text);
}
```

```
printf("File received successfully.\n");
fclose(file);
msgctl(msgid, IPC_RMID, NULL);
return 0;
}
```

OUTPUT:

```
[23A95A0515@Linux ~]$ vi receiver.c
[23A95A0515@Linux ~]$ cc receiver.c -o destin
[23A95A0515@Linux ~]$ ./destin
File received successfully.
```

WEEK-7

7.1) AIM: To design TCP iterative Client and server application to reverse the given input sentence.

SERVER PROGRAM:

```
import java.io.*;
import java.net.*;
public class ReverseServer
{
public static void main(String[] args)
{
int port = 1234;
try (ServerSocket serverSocket = new
ServerSocket(port)){ System.out.println("Server is running and waiting for
client connection..."); while (true)
{
Socket clientSocket = serverSocket.accept(); System.out.println("Client connected!");
BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
String input = in.readLine();
System.out.println("Received from client: " + input);
String reversed = new StringBuilder(input).reverse().toString(); out.println(reversed);
System.out.println("Sent to client: " + reversed); clientSocket.close();
System.out.println("Client connection closed.\n");
}
} catch (IOException e) {
System.out.println("Server exception: " + e.getMessage()); e.printStackTrace();
}
}
}
```

OUTPUT:

```
C:\Users\AEC_LAB_2_SYS_14\Desktop>javac ReverseServer.java

C:\Users\AEC_LAB_2_SYS_14\Desktop>java ReverseServer
Server is running and waiting for client connection...
Client connected!
Received from client: Hello
Sent to client: olleH
Client connection closed.
```

CLIENT PROGRAM:

```
import java.io.*; import java.net.*;
public class ReverseClient
{
public static void main(String[] args)
{
String hostname = "localhost"; int port = 1234;
try (Socket socket = new Socket(hostname, port))
{
BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));
PrintWriter out = new PrintWriter(socket.getOutputStream(), true); BufferedReader in = new
BufferedReader(new
InputStreamReader(socket.getInputStream())); System.out.print("Enter a sentence to reverse: ");
String sentence = userInput.readLine(); out.println(sentence);
String reversed = in.readLine();
System.out.println("Reversed sentence from server: " + reversed);

} catch (IOException e)
{
System.out.println("Client exception: " + e.getMessage()); e.printStackTrace();
}
}
}
```

OUTPUT:


```
Enter a sentence to reverse: Hello
Reversed sentence from server: olleH

C:\Users\AEC_LAB_2_SYS_14\Desktop>
```

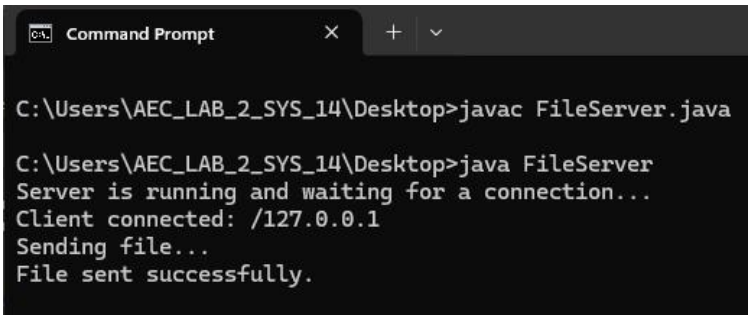
7.2) AIM: To design TCP client and server application to transfer file.

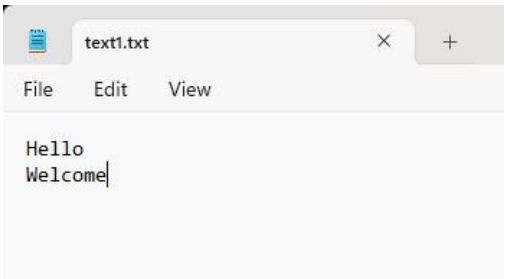
PROGRAM:

SERVER PROGARM:

```
import java.io.*; import java.net.*;
public class FileServer {
public static void main(String[] args) { int port = 5000;
String filePath = "C://Users/AEC_LAB_2_SYS_14/Desktop/text1.txt";
try (ServerSocket serverSocket = new ServerSocket(port))
{
System.out.println("Server is running and waiting for a connection...");
Socket clientSocket = serverSocket.accept();
System.out.println("Client connected: " + clientSocket.getInetAddress());
try (FileInputStream fileInputStream = new FileInputStream(filePath);
BufferedInputStream bufferedInputStream = new BufferedInputStream(fileInputStream);
OutputStream outputStream = clientSocket.getOutputStream())
{
byte[] buffer = new byte[4096]; int bytesRead;
System.out.println("Sending file...");
while ((bytesRead = bufferedInputStream.read(buffer)) != -1)
{
outputStream.write(buffer, 0, bytesRead);
}
System.out.println("File sent successfully.");
}
clientSocket.close();
} catch (IOException e) { e.printStackTrace();
}
}
}
```

OUTPUT:



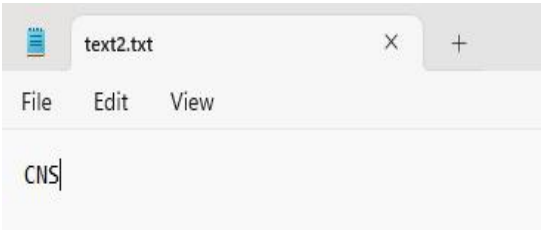


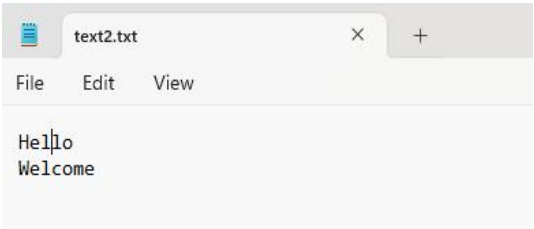
CLENT PROGRAM:

```
import java.io.*; import java.net.*;
public class FileClient {
public static void main(String[] args)
{
String serverAddress = "127.0.0.1"; int port = 5000;
String saveFilePath = "C://Users/AEC_LAB_2_SYS_14/Desktop/text2.txt";
 try (Socket socket = new Socket(serverAddress, port);
InputStream inputStream = socket.getInputStream();
FileOutputStream fileOutputStream = new FileOutputStream(saveFilePath);
BufferedOutputStream bufferedOutputStream = new BufferedOutputStream(fileOutputStream))
{
byte[] buffer = new byte[4096]; int bytesRead;
System.out.println("Receiving file...");
while ((bytesRead = inputStream.read(buffer)) != -1)
{
bufferedOutputStream.write(buffer, 0, bytesRead);
}
System.out.println("File received and saved to: " + saveFilePath);
} catch (IOException e) { e.printStackTrace();
}
}
}
```

OUTPUT:


```
File received and saved to: C://Users/AEC_LAB_2_SYS_14/Desktop/text2.t
xt
```





EXPERIMENT-8

8.1) Design a TCP concurrent server to convert a given text into upper case using multiplexing system call "select".

PROGRAM:

TCPConcurrentServerUpper.java:-

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.channels.*;
import java.net.*;
import java.util.*;
public class TCPConcurrentServerUpper {
   public static void main(String[] args) throws IOException {
      Selector selector = Selector.open();
      ServerSocketChannel server = ServerSocketChannel.open();
      server.bind(new InetSocketAddress(5000));
      server.configureBlocking(false);
      server.register(selector, SelectionKey.OP_ACCEPT);
         System.out.println("Server running....");
      while (true) {
         selector.select();
         for (Iterator<SelectionKey> it = selector.selectedKeys().iterator(); it.hasNext(); )
         {
            SelectionKey key = it.next(); it.remove();
            if (key.isAcceptable()) {
               SocketChannel client = server.accept();
               client.configureBlocking(false);
               client.register(selector, SelectionKey.OP_READ);
            } else if (key.isReadable()) {
               SocketChannel client = (SocketChannel) key.channel();
               ByteBuffer buffer = ByteBuffer.allocate(1024);
               if (client.read(buffer) == -1) { client.close(); continue; }
               buffer.flip();
               client.write(ByteBuffer.wrap(new String(buffer.array(), 0,
buffer.limit()).toUpperCase().getBytes()));
}}}}}
```

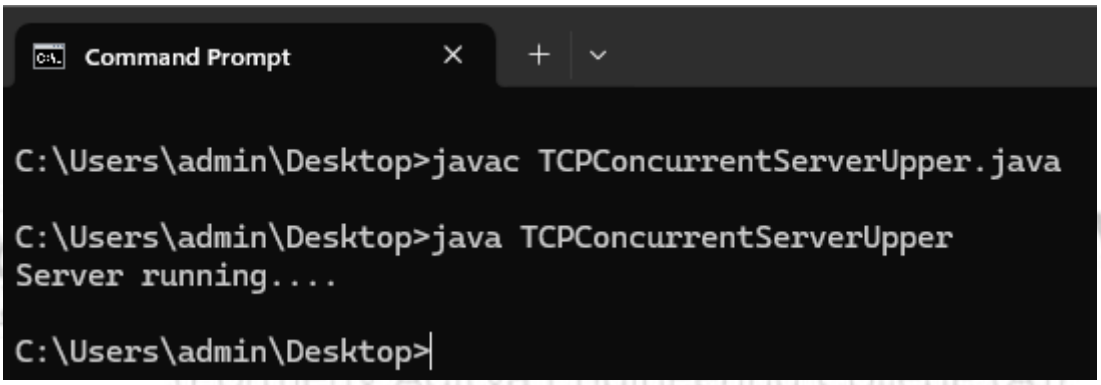
TCPClientUpper.java:-

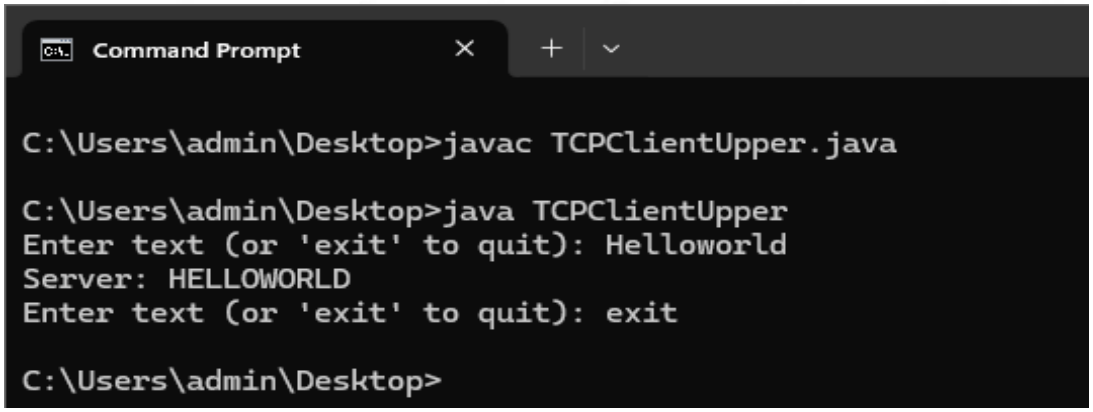
```
import java.io.IOException;
import java.net.InetSocketAddress;
import java.nio.ByteBuffer;
import java.nio.channels.SocketChannel;
import java.util.Scanner;
public class TCPClientUpper {
   public static void main(String[] args) throws IOException {
      SocketChannel client = SocketChannel.open(new InetSocketAddress("localhost",
5000));
      ByteBuffer buffer = ByteBuffer.allocate(1024);
      Scanner scanner = new Scanner(System.in);
      while (true) {
         System.out.print("Enter text (or 'exit' to quit): ");
```

```
            String msg = scanner.nextLine();
            if (msg.equalsIgnoreCase("exit"))

            break;
            buffer.clear();
            buffer.put(msg.getBytes());
            buffer.flip();
            client.write(buffer);
            buffer.clear();
            client.read(buffer);
            buffer.flip();
            System.out.println("Server: " + new String(buffer.array(), 0, buffer.limit()));
        }
        client.close();
        scanner.close();
    }}
```

OUTPUT:





8.2) Design a TCP concurrent server to echo given set of sentences using poll functions.

PROGRAM:

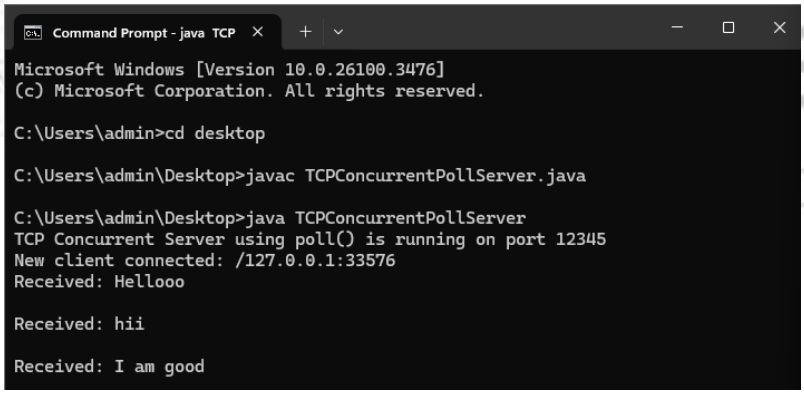
TCPConcurrentPollServer.java:-

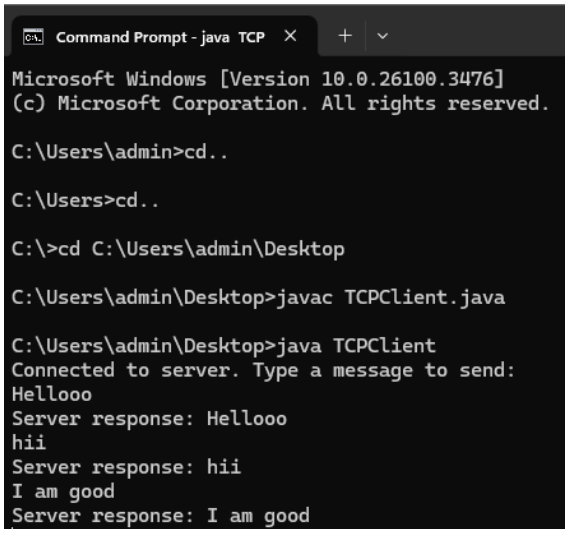
```
import java.io.IOException;
import java.net.InetSocketAddress;
import java.nio.ByteBuffer;
import java.nio.channels.*;
import java.util.Iterator;
import java.util.Set;
public class TCPConcurrentPollServer {
   private static final int PORT = 12345;
   private static final int BUFFER_SIZE = 1024;
   public static void main(String[] args) {
      try {
         ServerSocketChannel serverSocketChannel = ServerSocketChannel.open();
         serverSocketChannel.bind(new InetSocketAddress(PORT));
         serverSocketChannel.configureBlocking(false);
         Selector selector = Selector.open();
         serverSocketChannel.register(selector, SelectionKey.OP_ACCEPT);
         System.out.println("TCP Concurrent Server using poll() is running on port " + PORT);
         while (true) {
            selector.select();
            Set<SelectionKey> selectedKeys = selector.selectedKeys();
            Iterator<SelectionKey> iterator = selectedKeys.iterator();
            while (iterator.hasNext()) {
               SelectionKey key = iterator.next();
               iterator.remove();
               if (key.isAcceptable()) {
                  ServerSocketChannel serverChannel = (ServerSocketChannel) key.channel();
                  SocketChannel clientChannel = serverChannel.accept();
                  clientChannel.configureBlocking(false);
                  clientChannel.register(selector, SelectionKey.OP_READ);
                  System.out.println("New client connected: " +
clientChannel.getRemoteAddress());
               } else if (key.isReadable()) {
                  SocketChannel clientChannel = (SocketChannel) key.channel();
                  ByteBuffer buffer = ByteBuffer.allocate(BUFFER_SIZE);
                  int bytesRead = clientChannel.read(buffer);
                  if (bytesRead == -1) {
                     clientChannel.close();
                     System.out.println("Client disconnected");
                  } else {
                     buffer.flip();
                     String message = new String(buffer.array(), 0, bytesRead);
                     System.out.println("Received: " + message);
                     buffer.rewind();
                     clientChannel.write(buffer);
                  } }}}
      } catch (IOException e) {
         e.printStackTrace();
      }}}
```

TCPClient.java:-

```
import java.io.*;
import java.net.Socket;
public class TCPClient {
   public static void main(String[] args) {
      String serverIP = "localhost";
      int port = 12345;
      try (Socket socket = new Socket(serverIP, port);
          BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
          PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
          BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()))) {
         System.out.println("Connected to server. Type a message to send:");
         while (true) {
            String userMessage = input.readLine();
            if (userMessage.equalsIgnoreCase("exit")) break;
            out.println(userMessage);
            System.out.println("Server response: " + in.readLine());
         }
      } catch (IOException e) {
         e.printStackTrace();
      }}}
```

OUTPUT:





WEEK-9

9.1) AIM : Design UDP Client and server application to reverse the given input sentence

PROGRAM :

UDPClient.java

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;
public class UDPClient{
public static void main(String[] args)
{ try {
DatagramSocket clientSocket = new DatagramSocket();
InetAddress serverAddress = InetAddress.getByName("localhost");
int serverPort = 9876;
byte[] sendBuffer;
byte[] receiveBuffer = new byte[1024];
Scanner scanner = new Scanner(System.in);
System.out.println("Enter a sentence to reverse (or type 'exit' to quit):");
while (true) {
String sentence = scanner.nextLine();
if (sentence.equalsIgnoreCase("exit"))
{ System.out.println("Client exiting...");
break; }
sendBuffer = sentence.getBytes();
DatagramPacket sendPacket = new DatagramPacket(sendBuffer, sendBuffer.length, serverAddress,
serverPort);
clientSocket.send(sendPacket);
DatagramPacket receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);
clientSocket.receive(receivePacket);
String reversedSentence = new String(receivePacket.getData(), 0, receivePacket.getLength());
System.out.println("Reversed sentence from server: " + reversedSentence);  }
scanner.close();
clientSocket.close(); }
catch (Exception e)
{ e.printStackTrace();
} }}
```

OUTPUT :

```
Command Prompt - java UDPClient
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

C:\Users\admin>cd downloads

C:\Users\admin\Downloads>javac UDPClient.java

C:\Users\admin\Downloads>java UDPClient
Enter a sentence to reverse (or type 'exit' to quit):
hello goodmorning
Reversed sentence from server: gninromdoog olleh
```

UDPServer.java :

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
public class UDPServer {
public static void main(String[] args)
{ try {
DatagramSocket serverSocket = new DatagramSocket(9876);
System.out.println("Server is running...");
byte[] receiveBuffer = new byte[1024];
byte[] sendBuffer;
while (true) {
DatagramPacket receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);
serverSocket.receive(receivePacket);
String sentence = new String(receivePacket.getData(), 0, receivePacket.getLength());
System.out.println("Received from client: " + sentence);
String reversedSentence = new StringBuilder(sentence).reverse().toString();
sendBuffer = reversedSentence.getBytes();
DatagramPacket sendPacket = new DatagramPacket(sendBuffer, sendBuffer.length,
receivePacket.getAddress(), receivePacket.getPort());
serverSocket.send(sendPacket);
System.out.println("Reversed sentence sent to client: " + reversedSentence); }
} catch (Exception e)
{ e.printStackTrace();  }
}}
```

OUTPUT :


```
C:\Users\admin\Downloads>java UDPServer.java
Error: Could not find or load main class UDPServer.java

C:\Users\admin\Downloads>javac UDPServer.java

C:\Users\admin\Downloads>java UDPServer
Server is running...
Received from client: hello goodmorning
Reversed sentence sent to client: gninromdoog olleh
```

9.2) AIM : Design UDP Client server to transfer a file

PROGRAM :

UDPFileClient.java

```
import java.io.File;
import java.io.FileInputStream;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;
public class UDPFileClient {
public static void main(String[] args)
{ try {
DatagramSocket socket = new DatagramSocket();
InetAddress serverAddress = InetAddress.getByName("localhost");
int serverPort = 9876;
Scanner scanner = new Scanner(System.in);
System.out.print("Enter the file path to send: ");
String filePath = scanner.nextLine();
File file = new File(filePath);
if (!file.exists()) {
System.out.println("File does not exist.");
return;}
byte[] fileNameBytes = file.getName().getBytes();
DatagramPacket namePacket = new DatagramPacket(fileNameBytes, fileNameBytes.length,
serverAddress, serverPort);
socket.send(namePacket);
try (FileInputStream fis = new FileInputStream(file))
{ byte[] buffer = new byte[1024];
int bytesRead;
while ((bytesRead = fis.read(buffer)) != -1) {
DatagramPacket filePacket = new DatagramPacket(buffer, bytesRead, serverAddress, serverPort);
socket.send(filePacket); } }
byte[] endSignal = "EOF".getBytes();
DatagramPacket endPacket = new DatagramPacket(endSignal, endSignal.length, serverAddress,
serverPort);
socket.send(endPacket);
System.out.println("File sent successfully.");
socket.close(); }
catch (Exception e)
{ e.printStackTrace(); }}}
```

OUTPUT :

```
C:\Users\admin>cd downloads

C:\Users\admin\Downloads>javac UDPFileClient.java

C:\Users\admin\Downloads>java UDPFileClient
Enter the file path to send: C:\Users\admin\Downloads\UDPFileClient.class
File sent successfully.
```

UDPFileServer.java

```
import java.io.FileOutputStream;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
public class UDPFileServer {
public static void main(String[] args)
{ try {
DatagramSocket socket = new DatagramSocket(9876);
System.out.println("Server is running and waiting for file...");
byte[] buffer = new byte[1024];
DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
socket.receive(packet);
String fileName = new String(packet.getData(), 0, packet.getLength());
System.out.println("Receiving file: " + fileName);
try (FileOutputStream fos = new FileOutputStream("Received_" + fileName))
{ while (true) {
socket.receive(packet);
String endSignal = new String(packet.getData(), 0, packet.getLength());
if (endSignal.equals("EOF")) {
System.out.println("File transfer complete.");
break;
}
fos.write(packet.getData(), 0, packet.getLength());}}
socket.close();}
catch (Exception e)
{ e.printStackTrace(); }}}
```

OUTPUT :

```
C:\Users\admin\Downloads>javac UDPFileServer.java

C:\Users\admin\Downloads>java UDPFileServer
Server is running and waiting for file...
Receiving file: UDPFileClient.class
File transfer complete.
```

EXPERIMENT-4

4.1) Write a C Program to implement RSA algorithm.

PROGRAM:

```
#include<stdio.h>
#include<math.h>

//to find gcd
int gcd(int a, int b)
{
   if (b == 0) return a;
   return gcd(b, a % b);
}

int main()
{
   //2 random prime numbers
   double p = 3;
   double q = 7;
   double n=p*q;
   int count;
   double totient = (p-1)*(q-1);

   //public key
   //e stands for encrypt
   double e=6;

   //for checking co-prime which satisfies e>1
   while(e<totient)
   {
        count = gcd(e,totient);
        if(count==1)
        break;
        else
        e++;
   }

   //private key
   //d stands for decrypt
   double d,i;
   for(i=1;i<totient;i++)
   {
        if(fmod(e*i,totient)==1.0)
        {
                d=i; break;
                }
        }
```

```
    double msg = 12;
    double c = pow(msg,e);
    double m = pow(c,d);
    c=fmod(c,n);
    m=fmod(m,n);

    printf("\nMessage data = %lf",msg);
    printf("\np = %lf",p);
    printf("\nq = %lf",q);
    printf("\nn = pq = %lf",n);
    printf("\ntotient = %lf",totient);
    printf("\ne = %lf",e);
    printf("\nd = %lf",d);
    printf("\nEncrypted data = %lf",c);
    printf("\nOriginal Message Sent = %lf",m);

    return 0;
}
```

OUTPUT:

```
C:\Users\admin\Desktop\cns

Message data = 12.000000
p = 3.000000
q = 7.000000
n = pq = 21.000000
totient = 12.000000
e = 7.000000
d = 7.000000
Encrypted data = 12.000000
Original Message Sent = 14.000000
--------------------------------
Process exited after 0.03995 seconds with return value 0
Press any key to continue . . .
```

4.3) Write a C Program to implement Elgamal Cryptographic System.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
long long int mod_exp(long long int base, long long int exp, long long int mod) {
   long long int result = 1;
   while (exp > 0) {
      if (exp % 2 == 1) {
         result = (result * base) % mod;
      }
      exp = exp >> 1;
      base = (base * base) % mod;
   }
   return result;
}
int main() {
   long long int p = 11;  // A small prime number
   long long int e1 = 2;   // A primitive root modulo p
   long long int d = 3;   // Private key (random integer < p)
   long long int e2 = mod_exp(e1, d, p);  // Public key: e2 = e1^d mod p
   printf("Public Key: {p = %lld, e1 = %lld, e2 = %lld}\n", p, e1, e2);
   printf("Private Key: {d = %lld}\n", d);
   long long int M;
   printf("Enter message (integer < %lld): ", p);
   scanf("%lld", &M);
   if (M >= p) {
      printf("Error: Message must be smaller than p (%lld)\n", p);
      return 1;
   }
   long long int r = 3;  // Random integer (ephemeral key)
   long long int c1 = mod_exp(e1, r, p);
   long long int c2 = (M * mod_exp(e2, r, p)) % p;
   printf("Encrypted Message: {c1 = %lld, c2 = %lld}\n", c1, c2);
   long long int M_decrypted = (c2 * mod_exp(c1, p - 1 - d, p)) % p;
   printf("Decrypted Message: %lld\n", M_decrypted);
   return 0;
}
```

OUTPUT:

```
Public Key: {p = 11, e1 = 2, e2 = 8}
Private Key: {d = 3}
Enter message (integer < 11): 9
Encrypted Message: {c1 = 8, c2 = 10}
Decrypted Message: 9

--------------------------------
```

WEEK-10

10) Implement the following forms of IPC. a) Pipes b) FIFO

PROGRAM:

a) Pipes

//pipe.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main() {
    int pipefd[2], n;
    char buff[100];
    if (pipe(pipefd) == -1) {
        perror("Pipe creation failed");
        exit(1);
    }
    printf("\nRead fd = %d", pipefd[0]);
    printf("\nWrite fd = %d", pipefd[1]);
    // Write data into the pipe
    write(pipefd[1], "helloworld", 10);
    // Read from the read end of the pipe
    n = read(pipefd[0], buff, sizeof(buff));
    buff[n] = '\0'; // Null terminate the string
    printf("\nSize of the data: %d", n);
    printf("\nData from pipe: %s\n", buff);
    return 0;
}
```

OUTPUT:

```
[22A91A05K1@Linux ~]$ vi pipe.c
[22A91A05K1@Linux ~]$ gcc pipe.c
[22A91A05K1@Linux ~]$ ./a.out

Read fd = 3
Write fd = 4
Size of the data: 10
Data from pipe: helloworld
[22A91A05K1@Linux ~]$
```

b) FIFO

//fifoserver.c

```
#include<sys/stat.h>
#include<fcntl.h>
#include <unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
int main()
{
 int wrfd,rdfd,n,d,ret_val,count;
 char buf[50];
 /*create the first named pipe */
 ret_val=mkfifo("np1",0666);
 /*create the second named pipe */
 ret_val=mkfifo("np2",0666);
 /*open the first named pipe for reading*/
 rdfd=open("np1",O_RDONLY);
 /*open the second named pipe for writing*/
 wrfd=open("np2",O_WRONLY);
 /*read from the first pipe*/
 n=read(rdfd,buf,50);
 buf[n]='\0';//end of line
 printf("full duplex server:read from the pipe:%s\n",buf);
 /*convert the string to upper class*/
 count=0;
 while(count<n)
 {
 buf[count]=toupper(buf[count]);
 count++;
 }
 /*write the convertor string back to second pipe*/
write(wrfd,buf,strlen(buf));
return 0;
}
```

OUTPUT:

```
[22A91A05K1@Linux ~]$ vi fifoserver.c
[22A91A05K1@Linux ~]$ gcc fifoserver.c
[22A91A05K1@Linux ~]$ ./a.out
full duplex server:read from the pipe:hello world
[22A91A05K1@Linux ~]$
```

//fifoclient.c

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
int main() {
int wrfd, rdfd;
char msg[50] = "hello world", response[50];
 // Open pipes
wrfd = open("np1", O_WRONLY);
 rdfd = open("np2", O_RDONLY);
 // Write to server
 write(wrfd, msg, strlen(msg));
  printf("Client: Sent message - %s\n", msg);
  // Read response from server
    read(rdfd, response, sizeof(response));
    printf("Client: Received response - %s\n", response);
    return 0;
}
```

OUTPUT:

```
[22A91A05K1@Linux ~]$ vi fifoclient.c
[22A91A05K1@Linux ~]$ gcc fifoclient.c
[22A91A05K1@Linux ~]$ ./a.out
Client: Sent message - hello world
Client: Received response - HELLO WORLD
[22A91A05K1@Linux ~]$
```

Experiment-11

11) Implement file transfer using Message Queue form of IPC.

PROGRAM:

Sender:-

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define MSG_SIZE 256
#define FILE_PATH "input.txt"
#define QUEUE_KEY 1234
struct msg_buffer {
   long msg_type;
   char msg_text[MSG_SIZE];
};
int main() {
   key_t key = QUEUE_KEY;
   int msgid;
   struct msg_buffer message;
   FILE *file;
   msgid = msgget(key, 0666 | IPC_CREAT);
   if (msgid == -1) {
      perror("msgget");
      exit(EXIT_FAILURE);
   }
   file = fopen(FILE_PATH, "r");
   if (file == NULL) {
      perror("fopen");
      exit(EXIT_FAILURE);
   }
   message.msg_type = 1;
   while (fgets(message.msg_text, MSG_SIZE, file) != NULL) {
      if (msgsnd(msgid, &message, sizeof(message.msg_text), 0) == -1)
         { perror("msgsnd");
         exit(EXIT_FAILURE);
      }}
   strcpy(message.msg_text, "EOF");
   msgsnd(msgid, &message, sizeof(message.msg_text), 0);
   printf("File sent successfully.\n");
   fclose(file);
   return 0;
}
```

OUTPUT:

```
[22A91A05K1@Linux ~]$ vi cns11sender.c
[22A91A05K1@Linux ~]$ cc -o cns11sender cns11sender.c
[22A91A05K1@Linux ~]$ ./cns11sender
File sent successfully.
[22A91A05K1@Linux ~]$
```

Receiver:-

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define MSG_SIZE 256
#define FILE_PATH "output.txt"
#define QUEUE_KEY 1234
struct msg_buffer {
   long msg_type;
   char msg_text[MSG_SIZE];
};
int main() {
   key_t key = QUEUE_KEY;
   int msgid;
   struct msg_buffer message;
   FILE *file;
   msgid = msgget(key, 0666 | IPC_CREAT);
   if (msgid == -1) {
      perror("msgget");
      exit(EXIT_FAILURE);
   }
   file = fopen(FILE_PATH, "w");
   if (file == NULL) {
      perror("fopen");
      exit(EXIT_FAILURE);
   }
   while (1) {
      if (msgrcv(msgid, &message, sizeof(message.msg_text), 1, 0) == -1)
         { perror("msgrcv");
         exit(EXIT_FAILURE);
      }
      if (strcmp(message.msg_text, "EOF") == 0)
         { break;
      }
      fprintf(file, "%s", message.msg_text);
   }
   printf("File received successfully.\n");
   fclose(file);
   msgctl(msgid, IPC_RMID, NULL);
   return 0;
}
```

OUTPUT:

```
[23A95A0515@Linux ~]$ vi receiver.c
[23A95A0515@Linux ~]$ cc receiver.c -o destin
[23A95A0515@Linux ~]$ ./destin
File received successfully.
```

Experiment-11

11) Implement file transfer using Message Queue form of IPC.

PROGRAM:

Sender:-

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define MSG_SIZE 256
#define FILE_PATH "input.txt"
#define QUEUE_KEY 1234
struct msg_buffer {
   long msg_type;
   char msg_text[MSG_SIZE];
};
int main() {
   key_t key = QUEUE_KEY;
   int msgid;
   struct msg_buffer message;
   FILE *file;
   msgid = msgget(key, 0666 | IPC_CREAT);
   if (msgid == -1) {
      perror("msgget");
      exit(EXIT_FAILURE);
   }
   file = fopen(FILE_PATH, "r");
   if (file == NULL) {
      perror("fopen");
      exit(EXIT_FAILURE);
   }
   message.msg_type = 1;
   while (fgets(message.msg_text, MSG_SIZE, file) != NULL) {
      if (msgsnd(msgid, &message, sizeof(message.msg_text), 0) == -1)
         { perror("msgsnd");
         exit(EXIT_FAILURE);
      }}
   strcpy(message.msg_text, "EOF");
   msgsnd(msgid, &message, sizeof(message.msg_text), 0);
   printf("File sent successfully.\n");
   fclose(file);
   return 0;
}
```

OUTPUT:

```
[22A91A05K1@Linux ~]$ vi cns11sender.c
[22A91A05K1@Linux ~]$ cc -o cns11sender cns11sender.c
[22A91A05K1@Linux ~]$ ./cns11sender
File sent successfully.
[22A91A05K1@Linux ~]$
```

Receiver:-

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define MSG_SIZE 256
#define FILE_PATH "output.txt"
#define QUEUE_KEY 1234
struct msg_buffer {
   long msg_type;
   char msg_text[MSG_SIZE];
};
int main() {
   key_t key = QUEUE_KEY;
   int msgid;
   struct msg_buffer message;
   FILE *file;
   msgid = msgget(key, 0666 | IPC_CREAT);
   if (msgid == -1) {
      perror("msgget");
      exit(EXIT_FAILURE);
   }
   file = fopen(FILE_PATH, "w");
   if (file == NULL) {
      perror("fopen");
      exit(EXIT_FAILURE);
   }
   while (1) {
      if (msgrcv(msgid, &message, sizeof(message.msg_text), 1, 0) == -1)
         { perror("msgrcv");
         exit(EXIT_FAILURE);
      }
      if (strcmp(message.msg_text, "EOF") == 0)
         { break;
      }
      fprintf(file, "%s", message.msg_text);
   }
   printf("File received successfully.\n");
   fclose(file);
   msgctl(msgid, IPC_RMID, NULL);
   return 0;
}
```

OUTPUT:

```
[23A95A0515@Linux ~]$ vi receiver.c
[23A95A0515@Linux ~]$ cc receiver.c -o destin
[23A95A0515@Linux ~]$ ./destin
File received successfully.
```

EXPERIMENT-12

12) Design a RPC application to add and subtract a given pair of integers.

PROGRAM:

```
//Calculator.java
import java.rmi.Remote;
import java.rmi.RemoteException;
// Remote Interface
public interface Calculator extends Remote {
    int add(int a, int b) throws RemoteException;
    int subtract(int a, int b) throws RemoteException;
}

//CalculatorImpl.java
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
// Implementing the remote interface
public class CalculatorImpl extends UnicastRemoteObject implements Calculator {
  // Constructor
 protected CalculatorImpl() throws RemoteException {
   super();
 }
 // Implementation of add method
   public int add(int a, int b) throws RemoteException {
return a + b;
   }
   // Implementation of subtract method
 public int subtract(int a, int b) throws RemoteException {
   return a - b;
}
//CalculatorClient.java
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.Scanner;
public class CalculatorClient {
    public static void main(String[] args) {
     try {
// Locate the RMI registry
Registry registry = LocateRegistry.getRegistry("localhost", 1099);
// Lookup the remote object
Calculator stub = (Calculator) registry.lookup("CalculatorService");
// User input for numbers
Scanner scanner = new Scanner(System.in);
System.out.print("Enter first number: ");
int num1 = scanner.nextInt();
System.out.print("Enter second number: ");
int num2 = scanner.nextInt();
// Call remote methods
int sum = stub.add(num1, num2);
int difference = stub.subtract(num1, num2);
```

```
 System.out.println("Addition Result: " + sum);
System.out.println("Subtraction Result: " + difference);
 } catch (Exception e) {
   e.printStackTrace();
   }
  }
}
}

// CalculatorServer.java
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
public class CalculatorServer {
   public static void main(String[] args) {
      try {
         // Create remote object
         CalculatorImpl calc = new CalculatorImpl();

         // Start RMI registry (if not started already)
         Registry registry = LocateRegistry.createRegistry(1099);
         // Bind the remote object
         registry.rebind("CalculatorService", calc);
         System.out.println("Calculator Server is running...");
            } catch (Exception e) {
         e.printStackTrace();
            }
          }
         }
```

Output:

```
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\Users\admin\OneDrive\Desktop\exp-12>javac Calculator.java CalculatorImpl.
java CalculatorClient.java  CalculatorServer.java

C:\Users\admin\OneDrive\Desktop\exp-12>java CalculatorServer
Calculator Server is running...
```

```
C:\Users\admin\OneDrive\Desktop\exp-12>javac CalculatorClient.java

C:\Users\admin\OneDrive\Desktop\exp-12>java CalculatorClient
Enter first number: 78
Enter second number: 78
Addition Result: 156
Subtraction Result: 0

C:\Users\admin\OneDrive\Desktop\exp-12>java CalculatorClient
Enter first number: 98
Enter second number: 9
Addition Result: 107
Subtraction Result: 89
```