Name: Manasi Sanjay Kulkarni

Class: B.Tech IT – B

Roll No.: IT4228

EXPERIMENT NO. 1

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

int main(int argc, char *argv[])
{
        #pragma omp parallel
        {
                printf("Hello World! I'm thread %d out of %d threads.\n",
                omp_get_thread_num(), omp_get_num_threads());
        }
        return 0;
}
```

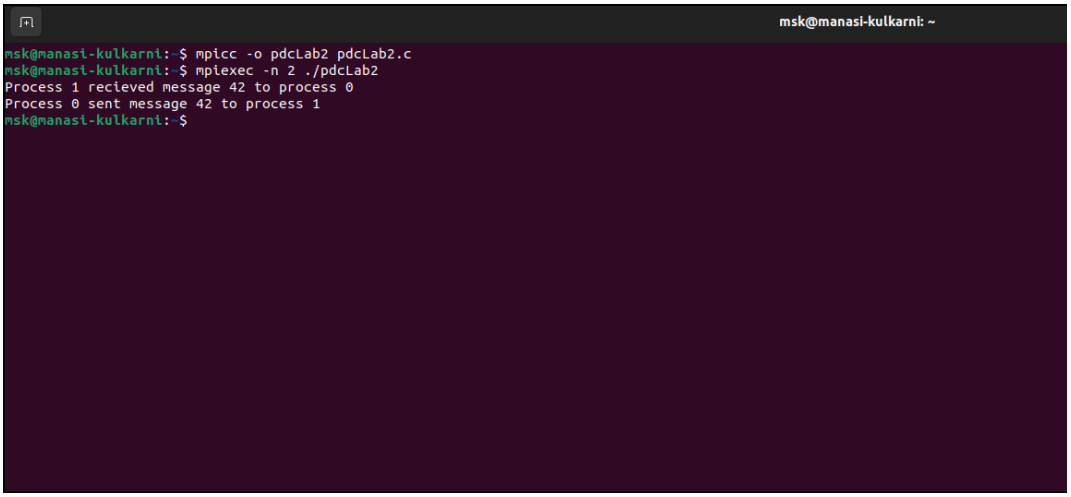
Name: Manasi Sanjay Kulkarni

Class: B.Tech IT – B

Roll No.: IT4228

EXPERIMENT NO. 2

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

int main(int argc, char *argv[])
{
	int rank, size;
	MPI_Init(&argc, &argv);
	MPI_Comm_rank(MPI_COMM_WORLD, &rank);
	MPI_Comm_size(MPI_COMM_WORLD, &size);

	if(rank==0)
	{
		int message = 42;
		MPI_Send(&message, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
		printf("Process %d sent message %d to process 1\n", rank, message);
	}
	else if(rank == 1)
	{
		int received_message;
		MPI_Recv(&received_message, 1, MPI_INT, 0, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
		printf("Process %d recieved message %d to process 0\n", rank, received_message);
	}

	MPI_Finalize();

	return 0;

}
```



Name: Manasi Sanjay Kulkarni

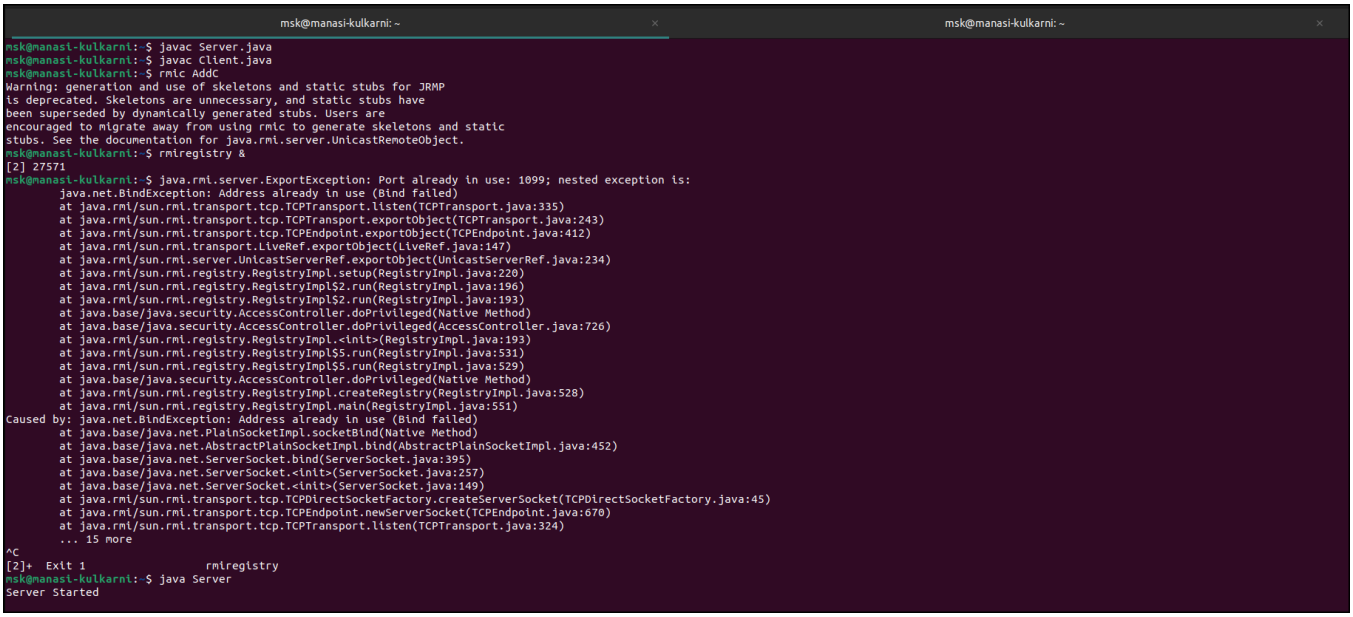
Class: B.Tech IT – B

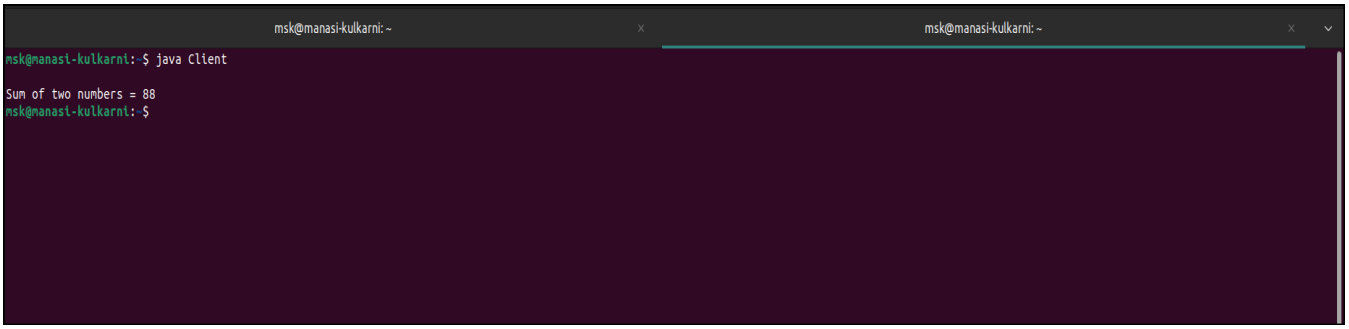
Roll No.: IT4228

EXPERIMENT NO. 3

```
//AddI.java
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface AddI extends Remote
{
	public int add(int x,int y) throws RemoteException;
}

//AddC.java
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class AddC extends UnicastRemoteObject implements AddI
{
	protected AddC()throws RemoteException
	{
		super();
	}
	public int add(int x,int y)
	{
		return x+y;
	}
}
```

```
//Server.java
import java.rmi.*;
import java.rmi.registry.*;

public class Server
{
	public static void main(String args[])throws Exception
	{
		AddC obj=new AddC();
		Naming.rebind("ADD",obj);
		System.out.println("Server Started");
	}
}

//Client.java
import java.rmi.*;
import java.util.Scanner;

public class Client
{
	public static void main(String args[])throws Exception
	{
		AddI AddIObj=(AddI)Naming.lookup("ADD");
		int res=AddIObj.add(24,64);
		System.out.println("\nSum of two numbers = "+res);
	}
}
```


msk@manasi-kulkarni: ~
msk@manasi-kulkarni: ~
msk@manasi-kulkarni:~$ javac Server.java
msk@manasi-kulkarni:~$ javac Client.java
msk@manasi-kulkarni:~$ rmic AddC
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.
msk@manasi-kulkarni:~$ rmiregistry &
[2] 27571
msk@manasi-kulkarni:~$ java.rmi.server.ExportException: Port already in use: 1099; nested exception is:
java.net.BindException: Address already in use (Bind failed)
at java.rmi/sun.rmi.transport.tcp.TCPTransport.listen(TCPTransport.java:335)
at java.rmi/sun.rmi.transport.tcp.TCPTransport.exportObject(TCPTransport.java:243)
at java.rmi/sun.rmi.transport.tcp.TCPEndpoint.exportObject(TCPEndpoint.java:412)
at java.rmi/sun.rmi.transport.LiveRef.exportObject(LiveRef.java:147)
at java.rmi/sun.rmi.server.UnicastServerRef.exportObject(UnicastServerRef.java:234)
at java.rmi/sun.rmi.registry.RegistryImpl.setup(RegistryImpl.java:228)
at java.rmi/sun.rmi.registry.RegistryImpl$2.run(RegistryImpl.java:196)
at java.rmi/sun.rmi.registry.RegistryImpl$2.run(RegistryImpl.java:193)
at java.base/java.security.AccessController.doPrivileged(Native Method)
at java.base/java.security.AccessController.doPrivileged(AccessController.java:726)
at java.rmi/sun.rmi.registry.RegistryImpl.<init>(RegistryImpl.java:193)
at java.rmi/sun.rmi.registry.RegistryImpl$5.run(RegistryImpl.java:531)
at java.rmi/sun.rmi.registry.RegistryImpl$5.run(RegistryImpl.java:529)
at java.base/java.security.AccessController.doPrivileged(Native Method)
at java.rmi/sun.rmi.registry.RegistryImpl.createRegistry(RegistryImpl.java:528)
at java.rmi/sun.rmi.registry.RegistryImpl.main(RegistryImpl.java:551)
Caused by: java.net.BindException: Address already in use (Bind failed)
at java.base/java.net.PlainSocketImpl.socketBind(Native Method)
at java.base/java.net.AbstractPlainSocketImpl.bind(AbstractPlainSocketImpl.java:452)
at java.base/java.net.ServerSocket.bind(ServerSocket.java:395)
at java.base/java.net.ServerSocket.<init>(ServerSocket.java:257)
at java.base/java.net.ServerSocket.<init>(ServerSocket.java:149)
at java.rmi/sun.rmi.transport.tcp.TCPDirectSocketFactory.createServerSocket(TCPDirectSocketFactory.java:45)
at java.rmi/sun.rmi.transport.tcp.TCPEndpoint.newServerSocket(TCPEndpoint.java:670)
at java.rmi/sun.rmi.transport.tcp.TCPTransport.listen(TCPTransport.java:324)
... 15 more
^C
[2]+ Exit 1 rmiregistry
msk@manasi-kulkarni:~$ java Server
Server Started


msk@manasi-kulkarni: ~
msk@manasi-kulkarni: ~
msk@manasi-kulkarni:~$ java Client
Sum of two numbers = 88
msk@manasi-kulkarni:~$

Name: Manasi Sanjay Kulkarni

Class: B.Tech IT – B

Roll No.: IT4228

EXPERIMENT NO. 4

```
import org.apache.activemq.ActiveMQConnectionFactory;
import javax.jms.*;
public class PubSubExample
{
	public static void main(String[] args) throws JMSException
	{
		// Connection parameters
		String brokerURL = "tcp://localhost:61616";
		String topicName = "exampleTopic";

		// Create a ConnectionFactory
		ConnectionFactory = new ActiveMQConnectionFactory(brokerURL);

		// Create and start a connection
		Connection = connectionFactory.createConnection();
		connection.start();

		// Create a session
		 Session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);

		// Create the topic
		Topic = session.createTopic(topicName);

		// Create a publisher
		MessageProducer publisher = session.createProducer(topic);
```

```
        // Create a subscriber
        MessageConsumer subscriber = session.createConsumer(topic);

        // Set up a listener for the subscriber
        subscriber.setMessageListener(message ->
        {
            if (message instanceof TextMessage)
            {
                try {
                    System.out.println("Received message: " + ((TextMessage)
                    message).getText());
                } catch (JMSException e) {
                    e.printStackTrace();
                }
            }
        });

        // Publish some messages
        sendMessage(session, publisher, "Hello, World!");
        sendMessage(session, publisher, "This is a test message.");
        sendMessage(session, publisher, "Goodbye!");

        // Clean up
        session.close();
        connection.close();
}

  private static void sendMessage(Session session, MessageProducer producer, String text)
 {
        try {
            TextMessage message = session.createTextMessage();
```

```
                    message.setText(text);
                    producer.send(message);
                    System.out.println("Sent message: " + text);
                } catch (JMSException e) {
                    e.printStackTrace();
                }
        }
}
```

```
Sent message: Hello, World!
Received message: Hello, World!
Sent message: This is a test message.
Received message: This is a test message.
Sent message: Goodbye!
Received message: Goodbye!
```

Name: Manasi Sanjay Kulkarni

Class: B.Tech IT – B

Roll No.: IT4228

EXPERIMENT NO. 5

Code for Web Service:

```
//Service resource.java
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/service")
public class ServiceResource
{
        @GET
        @Produces(MediaType.TEXT_PLAIN)
        public String getHello()
        {
                return "Hello from the web service!";
        }
}

//serviceApplications.java
import org.glassfish.jersey.server.ResourceConfig;
import javax.ws.rs.ApplicationPath;
@ApplicationPath("/api")
public class ServiceApplication extends ResourceConfig
{
        public ServiceApplication()
      {
```

```
        packages("your.package.name"); // Change this to your actual package name
    }
}
```

Code for Client application:

```
//ServiceClient.java
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.core.MediaType;
public class ServiceClient
{
    public static void main(String[] args)
    {
        Client = ClientBuilder.newClient();
        String serviceURL = "http://localhost:8080/api/service"; // Change the URL
            if needed
        String response =
    client.target(serviceURL).request(MediaType.TEXT_PLAIN).get(String.class);
        System.out.println("Response from the web service:");
        System.out.println(response);
        client.close();
    }
}
```

```
Response from the web service:
Hello from the web service!
```

Name: Manasi Sanjay Kulkarni

Class: B.Tech IT – B

Roll No.: IT4228

EXPERIMENT NO. 6

```
//ReverseServer.java
import ReverseModule.Reverse;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
class ReverseServer
{
        public static void main(String[] args)
        {
                try{
                        org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);
                        POA rootPOA =
                        POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
                        rootPOA.the_POAManager().activate();
                        ReverseImpl rvr = new ReverseImpl();
                        org.omg.CORBA.Object ref = rootPOA.servant_to_reference(rvr);
                        System.out.println("Step1");
                        Reverse h_ref = ReverseModule.ReverseHelper.narrow(ref);
                        System.out.println("Step2");
                        org.omg.CORBA.Object objRef =
                        orb.resolve_initial_references("NameService");
                        System.out.println("Step3");
                        NamingContextExt ncRef =
                        NamingContextExtHelper.narrow(objRef);
                        System.out.println("Step4");
```

```
                String name = "Reverse";
                NameComponent path[] = ncRef.to_name(name);
                ncRef.rebind(path,h_ref);
                System.out.println("Reverse Server reading and waiting....");
                orb.run();
            }
             catch(Exception e){
                e.printStackTrace();
            }
        }
}

//ReverseModule.idlj
module ReverseModule
{
        interface Reverse
        {
            string reverse_string(in string str);
        };
};

//ReverseClient.java
import ReverseModule.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import java.io.*;
class ReverseClient
{
        public static void main(String args[])
        {
```

```
		Reverse ReverseImpl=null;
		try {
			org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);
			org.omg.CORBA.Object objRef =
			orb.resolve_initial_references("NameService");
			NamingContextExt ncRef =
			NamingContextExtHelper.narrow(objRef);
			String name = "Reverse";
			ReverseImpl = ReverseHelper.narrow(ncRef.resolve_str(name));
			System.out.println("Enter String=");
			BufferedReader br = new BufferedReader(new
			InputStreamReader(System.in));
			String str= br.readLine();
			String tempStr= ReverseImpl.reverse_string(str);
			System.out.println(tempStr);
		}
		catch(Exception e) {
			e.printStackTrace();
		}
	}
}

//ReverseImpl.java
import ReverseModule.ReversePOA;
import java.lang.String;
class ReverseImpl extends ReversePOA
{
	ReverseImpl()
	{
		super();
		System.out.println("Reverse Object Created");
```

```
    }
    public String reverse_string(String name)
    {
        StringBuffer str=new StringBuffer(name);
        str.reverse();
        return (("Server Sends : "+str));
    }
}
```