PRACTICAL JOURNAL

CLOUD COMPUTING
&
SOFT COMPUTING TECHNIQUES

SUBMITTED BY
NAME
Seat NO

IN PARTIAL FULFILLMENT OF
M.Sc. (IT) Part-I (Sem-1)

UNIVERSITY OF MUMBAI
DEPARTMENT OF COMPUTER SCIENCE & IT
S.K.SOMAIYA COLLEGE OF ARTS, SCIENCE & COMMERCE
Vidyavihar (E), Mumbai-400077
2019-2020

CLOUD COMPUTING [PSIT1P3]

S K SOMAIYA COLLEGE OF ARTS, SCIENCE & COMMERCE
"Aurobindo"
Vidyavihar(East) Mumbai 400077

CERTIFICATE

This is to Certify that,
Mr./Ms. _____
Student of **M.Sc. I.T Part-I (Sem-I)** with seat no _____ and college enrolled roll no _____ has satisfactorily completed the practical in Information Technology Laboratory for the course **(PSIT103) Cloud Computing** in the program of **INFORMATION TECHNOLOGY** from the **UNIVERSITY OF MUMBAI** for the academic year 2019-2020.

(Subject In-Charge) **(HOD)**

(Examiners)

_____

INDEX

Practical No.	Practical Title	Date	Signature
1.	**Write a program for implementing Client Server communication model using TCP.** A) A client server based program using TCP to find if the number entered is prime. B) A client server TCP based chatting application.	16/11/19	
2.	**Write a program for implementing Client Server communication model using UDP.** A) A client server based program using UDP to find if the number entered is even or odd. B) A client server based program using UDP to find the factorial of the entered number. C) A program to implement simple calculator operations like addition, subtraction, multiplication and division. D) A program that finds the square, square root, cube and cube root of the entered number.	23/11/19	
3.	A multicast Socket example.	30/11/19	
4.	**Write a program to show the object communication using RMI.** A) A RMI based application program to display current date and time. B) A RMI based application program that converts digits to words, e.g. 123 will be converted to one two three.	30/11/19	
5.	Show the implementation of web services.	07/12/19	
6.	Implement Xen virtualization and manage with Xen Center	14/12/19	
7.	Implement virtualization using VMWare ESXi Server and managing with vCenter	14/12/19	
8.	Implement Windows Hyper V virtualization	21/12/19	
9.	Develop application for Microsoft Azure.	22/12/19	
10.	Develop application for Google App Engine	22/12/19	

Practical No: 01

Aim: Write a program for implementing Client Server communication model using TCP.

Practical 1A: A client server based program using TCP to find if the number entered isprime.

Code

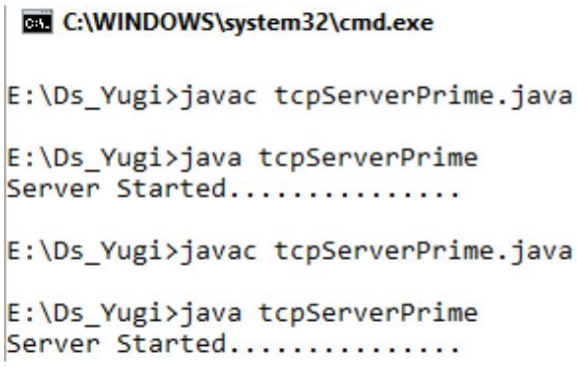
1. **tcpServerPrime.java**

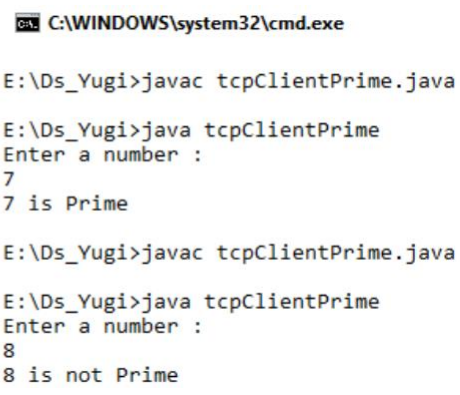
```
import java.net.*;
import java.io.*;
classtcpServerPrime
{
	public static void main(String args[])
	{
		try
		{
			ServerSocketss = new ServerSocket(8001);
			System.out.println("Server Started.....");
			Socket s = ss.accept();
			DataInputStream in = new
			DataInputStream(s.getInputStream()); int x= in.readInt();
			DataOutputStreamotc = new
			DataOutputStream(s.getOutputStream()); int y = x/2;
			if(x ==1 || x ==2 || x ==3)
			{
				otc.writeUTF(x + "is Prime");
				System.exit(0);
			}
			for(inti=2; i<=y; i++)
			{
				if(x%i != 0)
				{
					otc.writeUTF(x + " is Prime");
				}
				else
				{
					otc.writeUTF(x + " is not Prime");
				}
			}
		}
		catch(Exception e)
		{
			System.out.println(e.toString());
		}
	}
}
```

2. **tcpClientPrime.java**

```
import java.net.*;
import java.io.*;
classtcpClientPrime
{
	public static void main(String args[])
	{
		try
		{
			Socket cs = new Socket("LocalHost",8001);
			BufferedReaderinfu = new BufferedReader(new
InputStreamReader(System.in));
			System.out.println("Enter a number : ");
			int a = Integer.parseInt(infu.readLine());
			DataOutputStream out = new
DataOutputStream(cs.getOutputStream());
			out.writeInt(a);
			DataInputStream in = new
			DataInputStream(cs.getInputStream());
			System.out.println(in.readUTF()); cs.close();
		}
		catch(Exception e)
		{
			System.out.println(e.toString());
		}
	}
}
```

Output:





Practical 1B: A client server TCP based chatting application.

Code:

1. **ChatServer.java**

```
import java.net.*;
import java.io.*;
classChatServer
{
	public static void main(String args[])
	{
		try
		{
			ServerSocketss = new ServerSocket(8000);
			System.out.println("Waiting for client to
			connect..."); Socket s = ss.accept();
			BufferedReaderbr = new BufferedReader(new
InputStreamReader(System.in));
			DataOutputStream out = new
			DataOutputStream(s.getOutputStream()); DataInputStream in = new
			DataInputStream(s.getInputStream()); String receive, send;
			while((receive = in.readLine()) != null)
			{
				if(receive.equals("STOP"))
					break;
				System.out.println("Client Says : "+receive);
				System.out.print("Server Says : ");
				send = br.readLine();
				out.writeBytes(send+"\n");
			}
			br.close();
			in.close();
			out.close();
			s.close();
		}
		catch(Exception e)
		{
			e.printStackTrace();
		}
	}
}
```

2. **ChatClient.java**

```
import java.net.*;
import java.io.*;
classChatClient

{
	public static void main(String args[])
```

```
	{
		try
		{
			Socket s = new Socket("Localhost",8000);
			BufferedReaderbr = new BufferedReader(new
InputStreamReader(System.in));
			DataOutputStream out = new
			DataOutputStream(s.getOutputStream()); DataInputStream in = new
			DataInputStream(s.getInputStream()); String msg;
			System.out.println("To stop chatting with server type
			STOP"); System.out.print("Client Says: "); while((msg =
			br.readLine()) != null)
			{
				out.writeBytes(msg+"\n");
				if(msg.equals("STOP"))
					break;
				System.out.println("Server Says : "+in.readLine());
				System.out.print("Client Says : ");
			}
			br.close();
			in.close();
			out.close();
			s.close();
		}
		catch(Exception e)
		{
			e.printStackTrace();
		}
	}
}
```

Output:

Server:

```
E:\Ds_Yugi>java ChatServer
Waiting for client to connect..
Client Says : Hii...
Server Says : Hello
Client Says : How are you?
Server Says : Fine, thank you..
```

Clien

```
E:\Ds_Yugi>java ChatClient
To stop chatting with server type STOP
Client Says: Hii...
Server Says : Hello
Client Says : How are you?
Server Says : Fine, thank you..
Client Says : STOP
```

Practical No: 02

Aim: Write a program for implementing Client Server communication model using UDP.

Practical 2A:A client server based program using UDP to find if the number entered is evenor odd.

Code:

1. **udpServerEO.java**

```
/*Program which finds entered number is even or odd */
import java.io.*;
import java.net.*;
public class udpServerEO
{
	public static void main(String args[])
	{
		try
		{
			DatagramSocket ds = new DatagramSocket(2000);
			byte b[] = new byte[1024];
			DatagramPacketdp = new DatagramPacket(b,b.length);
			ds.receive(dp);
			String str = new String(dp.getData(),0,dp.getLength());
			System.out.println(str);
			int a= Integer.parseInt(str);
			String s= new String();
			if (a%2 == 0)
				s = "Number is even";
			else
				s = "Number is odd";

			byte b1[] = new byte[1024];
			b1 = s.getBytes();
			DatagramPacket dp1 = new
DatagramPacket(b1,b1.length,InetAddress.getLocalHost(),1000);
			ds.send(dp1);
		}
		catch(Exception e)
		{
			e.printStackTrace();
		}
	}
}
```

2. **udpClientEO.java**

```
/*Program which finds entered number is even or odd*/
import java.io.*;
import java.net.*;
public class udpClientEO
{
	public static void main(String args[])
	{
		try
		{
			DatagramSocket ds = new DatagramSocket(1000);
			BufferedReaderbr = new BufferedReader(new
InputStreamReader(System.in));
			System.out.println("Enter a number : ");
			String num = br.readLine();
			byte b[] = new byte[1024];
			b=num.getBytes();
			DatagramPacketdp = new
DatagramPacket(b,b.length,InetAddress.getLocalHost(),2000);
			ds.send(dp);
			byte b1[] = new byte[1024];
			DatagramPacket dp1 = new
			DatagramPacket(b1,b1.length); ds.receive(dp1);
			String str = new String(dp1.getData(),0,dp1.getLength());
			System.out.println(str);
		}
		catch(Exception e)
		{
			e.printStackTrace();
		}
	}
}
```

Output:

```
C:\WINDOWS\system32\cmd.exe

E:\Ds_Yugi>javac udpClientEO.java

E:\Ds_Yugi>java udpClientEO
Enter a number :
24
Number is even

E:\Ds_Yugi>java udpClientEO
Enter a number :
11
Number is odd
```

```
C:\WINDOWS\system32\cmd.exe

E:\Ds_Yugi>javac udpServerEO.java

E:\Ds_Yugi>java udpServerEO
24

E:\Ds_Yugi>java udpServerEO
11
```

Practical 2B: A client server based program using UDP to find the factorial of the enterednumber.

Code:

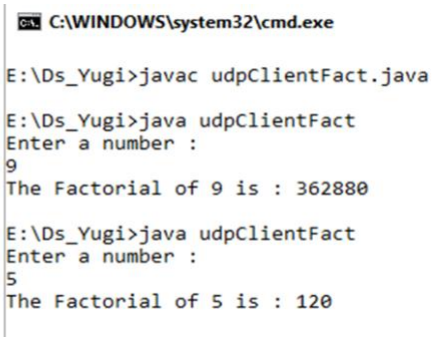
1. **udpServerFact.java**

```
/*Program which calculate factorial of a number*/
import java.io.*;
import java.net.*;
public class udpServerFact
{
	public static void main(String args[])
	{
		try
		{
			DatagramSocket ds = new DatagramSocket(2000);
			byte b[] = new byte[1024];
			DatagramPacketdp = new DatagramPacket(b,b.length);
			ds.receive(dp);
			String str = new String(dp.getData(),0,dp.getLength());
			System.out.println(str);
			int a= Integer.parseInt(str);
			int f = 1, i;
			String s= new String();
			for(i=1;i<=a;i++)
			{
				f=f*i;
			}
			s=Integer.toString(f);

			String str1 = "The Factorial of " + str + " is : " + f;
			byte b1[] = new byte[1024]; b1 = str1.getBytes();

			DatagramPacket dp1 = new
DatagramPacket(b1,b1.length,InetAddress.getLocalHost(),1000);
			ds.send(dp1);
		}
		catch(Exception e)
		{
			e.printStackTrace();
		}

	}
}
```

2. **udpClientFact.java**

```
/*Program which calculate factorial of a number*/

import java.io.*;
import java.net.*;
public class udpClientFact
{
	public static void main(String args[])
	{
		try
		{
			DatagramSocket ds = new DatagramSocket(1000);
			BufferedReaderbr = new BufferedReader(new
InputStreamReader(System.in));
			System.out.println("Enter a number : ");
			String num = br.readLine();
			byte b[] = new byte[1024];
			b=num.getBytes();
			DatagramPacketdp = new
DatagramPacket(b,b.length,InetAddress.getLocalHost(),2000);
			ds.send(dp);
			byte b1[] = new byte[1024];
			DatagramPacket dp1 = new
			DatagramPacket(b1,b1.length); ds.receive(dp1);
			String str = new String(dp1.getData(),0,dp1.getLength());
			System.out.println(str);
		}
		catch(Exception e)
		{
			e.printStackTrace();
		}
	}
}
```

Output:



Practical 2C: A program to implement simple calculator operations like addition,subtraction, multiplication and division.

Code:

1. **RPCServer.java**

```
importjava.util.*; import
java.net.*;
classRPCServer
{
        DatagramSocket  ds;
        DatagramPacketdp;
        String str,methodName,result;
        int val1,val2; RPCServer()
        {
                try
                {
                        ds=new DatagramSocket(1200);
                        byte b[]=new byte[4096];
                        while(true)
                        {
                                dp=new DatagramPacket(b,b.length);
                                ds.receive(dp);
                                str=new String(dp.getData(),0,dp.getLength());
                                if(str.equalsIgnoreCase("q"))
                                {
                                        System.exit(1);
                                }
                                else
                                {
                                        StringTokenizerst = new StringTokenizer(str," ");
                                        inti=0;
                                        while(st.hasMoreTokens())
                                        {
                                                String token=st.nextToken();
                                                methodName=token;
                                                val1 = Integer.parseInt(st.nextToken());
                                                val2 = Integer.parseInt(st.nextToken());
                                        }
                                }
                                System.out.println(str);
                                InetAddressia = InetAddress.getLocalHost();
                                if(methodName.equalsIgnoreCase("add"))
                                {
                                        result= "" + add(val1,val2);
                                }
                                else if(methodName.equalsIgnoreCase("sub"))
                                {
                                        result= "" + sub(val1,val2);
                                }
```

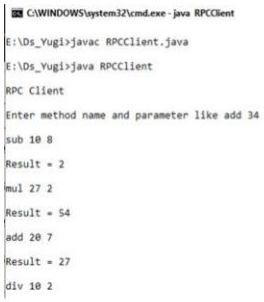
```
                                else if(methodName.equalsIgnoreCase("mul"))
                                {
                                        result= "" + mul(val1,val2);

                                }
                                else if(methodName.equalsIgnoreCase("div"))
                                {
                                        result= "" + div(val1,val2);
                                }
                                byte b1[]=result.getBytes();
                                DatagramSocket ds1 = new DatagramSocket();
                                DatagramPacket dp1 = new
DatagramPacket(b1,b1.length,InetAddress.getLocalHost(), 1300);
                                System.out.println("result :
                                "+result+"\n"); ds1.send(dp1);
                        }
                }
                catch (Exception e)
                {
                        e.printStackTrace();
                }
        }
        publicint add(int val1, int val2)
        {
                return val1+val2;
        }
        publicint sub(int val3, int val4)
        {
                return val3-val4;
        }

        publicintmul(int val3, int val4)
        {
                return val3*val4;
        }
        publicint div(int val3, int val4)
        {
                return val3/val4;
        }

        public static void main(String[] args)
        {
                newRPCServer();
        }
}
```

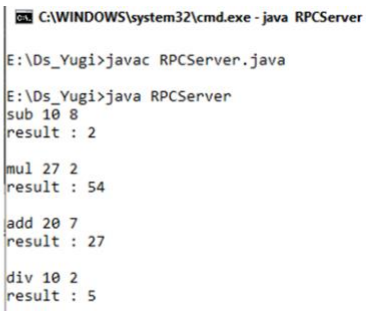
2. **RPCClient.java**

```
import java.io.*;
import java.net.*;
classRPCClient
```

```
{
	RPCClient()

	{
		try
		{
			InetAddressia = InetAddress.getLocalHost();
			DatagramSocket ds = new DatagramSocket();
			DatagramSocket ds1 = new DatagramSocket(1300);
			System.out.println("\nRPC Client\n");
			System.out.println("Enter method name and parameter like add 3
4\n");
			while (true)
			{

				BufferedReaderbr = new
BufferedReader(new InputStreamReader(System.in));

				String str = br.readLine();
				byte b[] = str.getBytes();
				DatagramPacketdp = new
DatagramPacket(b,b.length,ia,1200);
				ds.send(dp);
				dp = new DatagramPacket(b,b.length);
				ds1.receive(dp);
				String s = new String(dp.getData(),0,dp.getLength());
				System.out.println("\nResult = " + s + "\n");
			}
		}
		catch (Exception e)
		{
			e.printStackTrace();
		}
	}

	public static void main(String[] args)
	{
		newRPCClient();
	}
}
```

Output:





Practical 2D: A program that finds the square, square root, cube and cube root of the enterednumber.

Code:

1. **RPCNumServer.java**

```
importjava.util.*;
import java.net.*;
import java.io.*;
classRPCNumServer
 {
        DatagramSocket  ds;
        DatagramPacketdp;
        String str,methodName,result;
        intval;
        RPCNumServer()
        {
                try
                {
                        ds=new DatagramSocket(1200);
                        byte b[]=new byte[4096];
                        while(true)
                        {
                                dp=new DatagramPacket(b,b.length);
                                ds.receive(dp);
                                str=new
                                String(dp.getData(),0,dp.getLength());
                                if(str.equalsIgnoreCase("q")) {
                                        System.exit(1);
                                }
                                else
                                {
                                        StringTokenizerst = new StringTokenizer(str," ");
                                        inti=0;
                                        while(st.hasMoreTokens())
                                        {
                                                String token=st.nextToken();
                                                methodName=token;
                                                val = Integer.parseInt(st.nextToken());
                                        }
                                }
                                System.out.println(str);
                                InetAddressia = InetAddress.getLocalHost();
                                if(methodName.equalsIgnoreCase("square"))
                                {
                                        result= "" + square(val);
                                }
                                else if(methodName.equalsIgnoreCase("squareroot"))
                                {
                                        result= "" + squareroot(val);
                                }
```

```
                                else if(methodName.equalsIgnoreCase("cube"))
                                {
                                        result= "" + cube(val);

                                }
                                else if(methodName.equalsIgnoreCase("cuberoot"))
                                {
                                        result= "" + cuberoot(val);
                                }
                                byte b1[]=result.getBytes();
                                DatagramSocket ds1 = new DatagramSocket();
                                DatagramPacket dp1 = new
DatagramPacket(b1,b1.length,InetAddress.getLocalHost(), 1300);
                                System.out.println("result :
                                "+result+"\n"); ds1.send(dp1);
                        }
                }
                catch (Exception e)
                {
                        e.printStackTrace();
                }
        }
        public double square(int a) throws Exception
        {
                doubleans;
                ans = a*a;
                returnans;
        }
        public double squareroot(int a) throws Exception
        {
                doubleans;
                ans = Math.sqrt(a);
                returnans;
        }
        public double cube(int a) throws Exception
        {
                doubleans;
                ans = a*a*a;
                returnans;
        }
        public double cuberoot(int a) throws Exception
        {
                doubleans;
                ans = Math.cbrt(a);
                returnans;
        }
        public static void main(String[] args)
        {
                newRPCNumServer();
```

```
	}
}
```

2. **RPCNumClient.java**

```
import java.io.*;
import java.net.*;
classRPCNumClient
{
	RPCNumClient()
	{
		try
		{
			InetAddressia = InetAddress.getLocalHost();
			DatagramSocket ds = new DatagramSocket();
			DatagramSocket ds1 = new DatagramSocket(1300);
			System.out.println("\nRPC Client\n");
			System.out.println("1. Square of the number - square\n2. Square root
of the number - squareroot\n3. Cube of the number - cube\n4. Cube root of the number -
cuberoot");
			System.out.println("Enter method name and the number\n");
			while (true)
			{
				BufferedReaderbr = new
BufferedReader(new InputStreamReader(System.in));
				String str = br.readLine();
				byte b[] = str.getBytes();
				DatagramPacketdp = new
DatagramPacket(b,b.length,ia,1200);
				ds.send(dp);
				dp = new DatagramPacket(b,b.length);
				ds1.receive(dp);
				String s = new String(dp.getData(),0,dp.getLength());
				System.out.println("\nResult = " + s + "\n");
			}
		}
		catch (Exception e)
		{
			e.printStackTrace();
		}
	}
	public static void main(String[] args)
	{
		newRPCNumClient();
	}
}
```

Output:

```
C:\WINDOWS\system32\cmd.exe - java RPCNumServer

E:\Ds_Yugi>javac RPCNumServer.java

E:\Ds_Yugi>java RPCNumServer
square 7
result : 49.0

squareroot 25
result : 5.0

cube 3
result : 27.0

cuberoot 27
result : 3.0
```

```
C:\WINDOWS\system32\cmd.exe - java RPCNumClient

1. Square of the number - square
2. Square root of the number - squareroot
3. Cube of the number - cube
4. Cube root of the number - cuberoot
Enter method name and the number

square 7

Result = 49.0

squareroot 25

Result = 5.0

cube 3

Result = 27.0

cuberoot 27

Result = 3.0
```

Practical No: 03

Aim: A multicast Socket example.

Code:

1. **BroadcastServer.java**

```
import java.net.*;
import java.io.*;
importjava.util.*;
public class BroadcastServer
{
        public static final int PORT = 1234;
        public static void main(String args[])throws
        Exception {
                MulticastSocket  socket;
                DatagramPacket packet;
                InetAddress address;
                // set the multicast address to your local subnet
                address = InetAddress.getByName("239.1.2.3");
                socket = new MulticastSocket();
                // join a Multicast group and send the group
                messages socket.joinGroup(address);
                byte[] data =
                null; for(;;)
                {
                        Thread.sleep(10000);
                        System.out.println("Sending "); String
                        str = ("This is Neha Calling...."); data
                        = str.getBytes();
                        packet = new DatagramPacket(data, str.length(),address,PORT);
                        // Sends the packet
                        socket.send(packet);
                } // end for
        } // end main
 } // end class BroadcastServer
```

2. **BroadcastClient.java**

```
import java.net.*;
import java.io.*;
public class BroadcastClient
{
        public static final int PORT = 1234;
        public static void main(String args[])throws
        Exception {
                MulticastSocket socket;
                DatagramPacket packet;
                InetAddress address;
```

```
                // set the mulitcast address to your local subnet
                address = InetAddress.getByName("239.1.2.3");
                socket = new MulticastSocket(PORT);
                //join a Multicast group and wait for a message
                socket.joinGroup(address); byte[] data = new
                byte[100];
                packet = new DatagramPacket(data,data.length);
                for(;;)
                {
                        // receive the packets
                        socket.receive(packet);
                        String str = new String(packet.getData());
                        System.out.println("Message received from "+ packet.getAddress() + "
Message is : "+str);
                } // for
        } // main
} // end BroadcastClient
```

Output:

Practical No: 04

Aim: Write a program to show the object communication using RMI.

Practical 4A: A RMI based application program to display current date and time.
Code:

1. **InterDate.java**

```
importjava.rmi.*;
public interface InterDate extends Remote
{
        public String display() throws Exception;

}
```

2. **ServerDate.java**

```
importjava.rmi.*;
importjava.rmi.server.*;
importjava.util.*;
public class ServerDate extends UnicastRemoteObject implements
InterDate {
        publicServerDate() throws Exception
        {
        }
        public String display() throws Exception
        {
                String str = "";
                Date d = new Date();
                str = d.toString();
                returnstr;
        }
        public static void main(String args[]) throws
        Exception {
                ServerDate s1 = new ServerDate();
                Naming.bind("DS",s1);
                System.out.println("Object registered.....");
        }
}
```

3. **ClientDate.java**

```
importjava.rmi.*;
import java.io.*; public
class ClientDate
{
        public static void main(String args[]) throws
        Exception {
                String s1;
                InterDate h1 = (InterDate)Naming.lookup("DS");
                s1 = h1.display();
```

```
            System.out.println(s1);
        }
}
```

Output:

```
C:\WINDOWS\system32\cmd.exe - rmiregistry
E:\Ds_Yugi>javac ServerDate.java

E:\Ds_Yugi>javac ClientDate.java

E:\Ds_Yugi>rmic ServerDate
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.

E:\Ds_Yugi>rmiregistry
```

```
C:\WINDOWS\system32\cmd.exe - java ServerDate

E:\Ds_Yugi>java ServerDate
Object registered.....
```

```
C:\WINDOWS\system32\cmd.exe

E:\Ds_Yugi>java ClientDate
Thu Jan 04 17:38:00 IST 2018
```

Practical 4B: A RMI based application program that converts digits to words, e.g. 123 willbe converted to one two three.

Code:

1. **InterConvert.java**

```
importjava.rmi.*;
public interface InterConvert extends Remote
{
        public String convertDigit(String no) throws Exception;
}
```

2. **ServerConvert.java**

```
importjava.rmi.*;
importjava.rmi.server.*;
public class ServerConvert extends UnicastRemoteObject implements
InterConvert {
        publicServerConvert() throws Exception
        {
        }
```

```
public String convertDigit(String no) throws Exception
{
        String str = "";
        for(inti = 0; i<no.length(); i++)
        {
                int p = no.charAt(i);
                if( p == 48)
                {
                        str += "zero ";
                }
                if( p == 49)
                {
                        str += "one ";
                }
                if( p == 50)
                {
                        str += "two ";
                }
                if( p == 51)
                {
                        str += "three ";
                }
                if( p == 52)
                {
                        str += "four ";
                }
                if( p == 53)
                {
                        str += "five ";
                }
                if( p == 54)
                {
                        str += "six ";
                }
                if( p == 55)
                {
                        str += "seven ";
                }
                if( p == 56)
                {
                        str += "eight ";
                }
                if( p == 57)
                {
                        str += "nine ";
                }
        }
        returnstr;
}
```

```
        public static void main(String args[]) throws
        Exception {
                ServerConvert s1 = new ServerConvert();
                Naming.bind("Wrd",s1);
                System.out.println("Object registered....");
        }
}
```

3. **ClientConvert.java**

```
importjava.rmi.*;
import java.io.*;
public class ClientConvert
{
        public static void main(String args[]) throws
        Exception {
                InterConvert h1 = (InterConvert)Naming.lookup("Wrd");
                BufferedReaderbr = new BufferedReader(new
InputStreamReader(System.in));
                System.out.println("Enter a number : \t");
                String no = br.readLine();
                String ans = h1.convertDigit(no);
                System.out.println("The word representation of the entered digit is : " +ans);
        }
}
```

Output:

Practical No: 05

Aim: Show the implementation of web services.

What Are Web Services?

Web services are client and server applications that communicate over the World WideWeb's (WWW) HyperText Transfer Protocol (HTTP). As described by the World Wide Web Consortium (W3C), web services provide a standard means of interoperating between software applications running on a variety of platforms and frameworks. Web services are characterized by their great interoperability and extensibility, as well as their machine-processable descriptions, thanks to the use of XML. Web services can be combined in a loosely coupled way to achieve complex operations. Programs providing simple services can interact with each other to deliver sophisticated added-value services.

Types of Web Services:

On the conceptual level, a service is a software component provided through a network-accessible endpoint. The service consumer and provider use messages to exchange invocation request and response information in the form of self-containing documents that make very few assumptions about the technological capabilities of the receiver.

On a technical level, web services can be implemented in various ways. The two types of web services can be distinguished as "big" web services and "RESTful" web services.

1) **"Big" Web Services:**

In Java EE 6, JAX-WS provides the functionality for "big" web services. Big web services use XML messages that follow the Simple Object Access Protocol (SOAP) standard, an XML language defining a message architecture and message formats. Such systems often contain a machine-readable description of the operations offered by the service, written in the Web Services Description Language (WSDL), an XML language for defining interfaces syntactically.

The SOAP message format and the WSDL interface definition language have gained widespread adoption. Many development tools, such as NetBeans IDE, can reduce the complexity of developing web service applications.

A SOAP-based design must include the following elements.

- A formal contract must be established to describe the interface that the web service offers. WSDL can be used to describe the details of the contract, which may include messages, operations, bindings, and the location of the web service. You may also process SOAP messages in a JAX-WS service without publishing a WSDL.
- The architecture must address complex nonfunctional requirements. Many web service specifications address such requirements and establish a common vocabulary for them. Examples include transactions, security, addressing, trust, coordination, and so on.
- The architecture needs to handle asynchronous processing and invocation. In such cases, the infrastructure provided by standards, such as Web Services Reliable Messaging (WSRM), and APIs, such as JAX-WS, with their client-side asynchronous invocation support, can be leveraged out of the box.

2) **RESTful Web Services:**

In Java EE 6, JAX-RS provides the functionality for Representational State Transfer (RESTful) web services. REST is well suited for basic, ad hoc integration scenarios. RESTful web services, often better integrated with HTTP than SOAP-based services are, do not require XML messages or WSDL service–API definitions.

Project Jersey is the production-ready reference implementation for the JAX-RS specification. Jersey implements support for the annotations defined in the JAX-RS specification, making it easy for developers to build RESTful web services with Java and the Java Virtual Machine (JVM).

Because RESTful web services use existing well-known W3C and Internet Engineering Task Force (IETF) standards (HTTP, XML, URI, MIME) and have a lightweight infrastructure that allows services to be built with minimal tooling, developing RESTful web services is inexpensive and thus has a very low barrier for adoption. You can use a development tool such as NetBeans IDE to further reduce the complexity of developing RESTful web services.

A RESTful design may be appropriate when the following conditions are met.

- The web services are completely stateless. A good test is to consider whether the interaction can survive a restart of the server.

- A caching infrastructure can be leveraged for performance. If the data that the web service returns is not dynamically generated and can be cached, the caching infrastructure that web servers and other intermediaries inherently provide can be leveraged to improve performance. However, the developer must take care because such caches are limited to the HTTP GET method for most servers.
- The service producer and service consumer have a mutual understanding of the context and content being passed along. Because there is no formal way to describe the web services interface, both parties must agree out of band on the schemas that describe the data being exchanged and on ways to process it meaningfully. In the real world, most commercial applications that expose services as RESTful implementations also distribute so-called value-added toolkits that describe the interfaces to developers in popular programming languages.
- Bandwidth is particularly important and needs to be limited. REST is particularly useful for limited-profile devices, such as PDAs and mobile phones, for which the overhead of headers and additional layers of SOAP elements on the XML payload must be restricted.
- Web service delivery or aggregation into existing web sites can be enabled easily with a RESTful style. Developers can use such technologies as JAX-RS and Asynchronous JavaScript with XML (AJAX) and such toolkits as Direct Web Remoting (DWR) to consume the services in their web applications. Rather than starting from scratch, services can be exposed with XML and consumed by HTML pages without significantly refactoring the existing web site architecture. Existing developers will be more productive because they are adding to something they are already familiar with rather than having to start from scratch with new technology.

Deciding Which Type of Web Service to Use:

Basically, you would want to use RESTful web services for integration over the web and use big web services in enterprise application integration scenarios that have advanced quality of service (QoS) requirements.

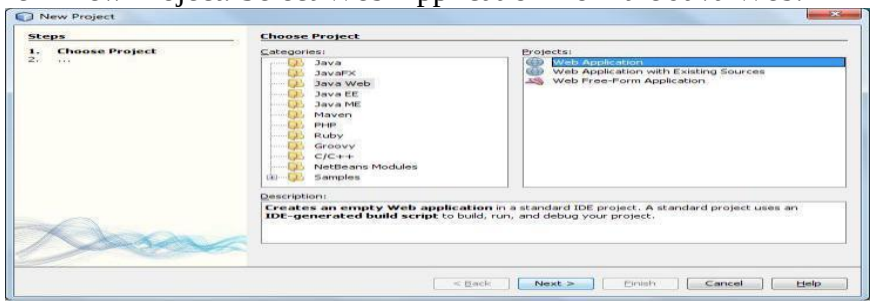
- **JAX-WS**: addresses advanced QoS requirements commonly occurring in enterprisecomputing. When **compared to JAX-RS, JAX-WS makes it easier to support the WS-* set of protocols, which provide standards for security and reliability, among other things, and interoperate with other WS-* conforming clients and** servers.
- **JAX-RS**: makes it easier to write web applications that apply some or all of theconstraints of the REST style to induce desirable properties in the application, such as loose coupling (evolving the server is easier without breaking existing clients), scalability (start small and grow), and architectural simplicity (use off-the-shelf components, such as proxies or HTTP routers). You would choose to use JAX-RS for your web application because it is easier for many types of clients to consume RESTful web services while enabling the server side to evolve and scale. Clients can choose to consume some or all aspects of the service and mash it up with other web-based services.

Practical 5A: Implementing “Big” Web Service.

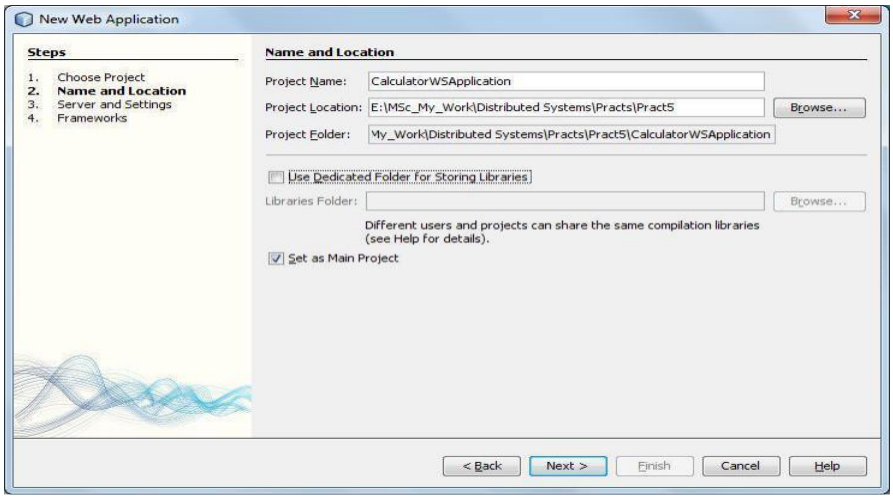
1) Creating a Web Service

A. Choosing a Container:

1. oose File > New Project. Select Web Application from the Java Web.



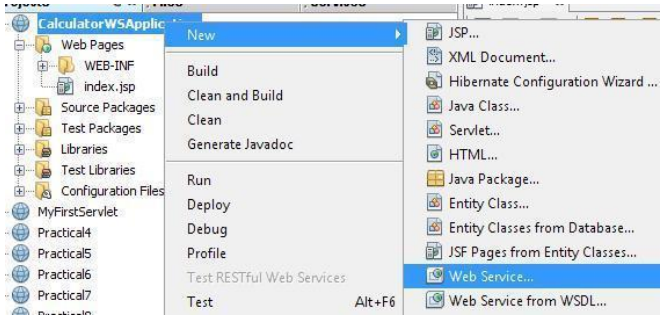
2. Name the project CalculatorWSApplication. Select a location for the project. Click Next.



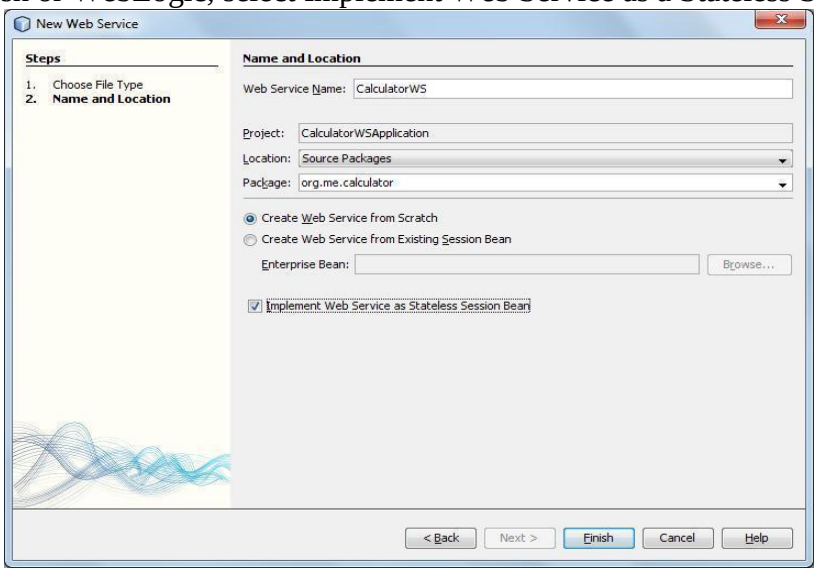
3. Select your server and Java EE version and click Finish.

B. **Creating a Web Service from a Java Class**

1. Right-click the CalculatorWSApplication node and choose New > Web Service.



2. Name the web service CalculatorWS and type org.me.calculator in Package. Leave Create Web Service from Scratch selected. If you are creating a Java EE 6 project on GlassFish or WebLogic, select Implement Web Service as a Stateless Session Bean.



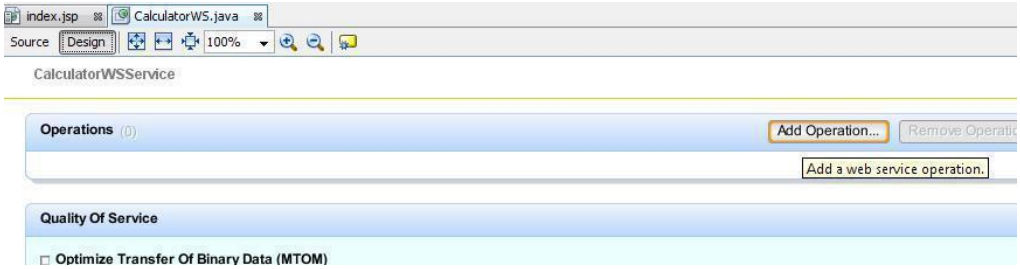
3. Click Finish. The Projects window displays the structure of the new web service and the source code is shown in the editor area.

2) **Adding an Operation to the Web Service**

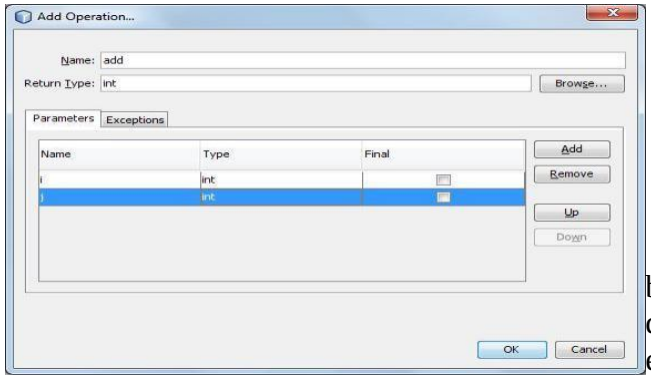
The goal of this exercise is to add to the web service an operation that adds two numbers received from a client. The NetBeans IDE provides a dialog for adding an operation to a web service. You can open this dialog either in the web service visual designer or in the web service context menu.

A. **To add an operation to the web service:**

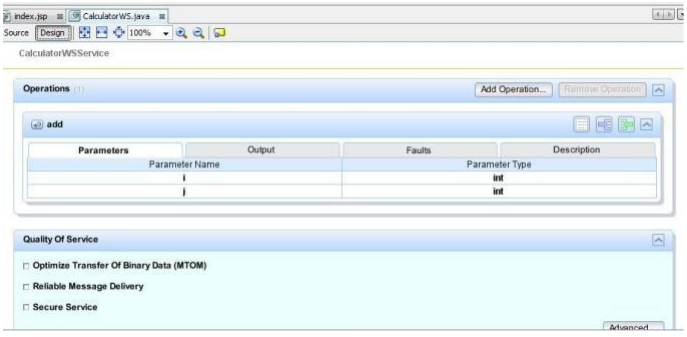
1. Change to the Design view in the editor.



2. Click Add Operation in either the visual designer or the context menu. The Add Operation dialog opens.
3. In the upper part of the Add Operation dialog box, type add in Name and type int in the Return Type drop-down list.
4. In the lower part of the Add Operation dialog box, click Add and create a parameter of type int named i.
5. Click Add again and create a parameter of type int called j. You now see the following:



6. Click OK at the bottom of the Add Operation dialog box. You return to the editor.
7. The visual designer now displays the following:



8. Click Source. And code the following.

```
@WebMethod(operationName = "add")
   publicint add(@WebParam(name = "i") inti, @WebParam(name = "j") int j)
{
        int k = i + j;
```

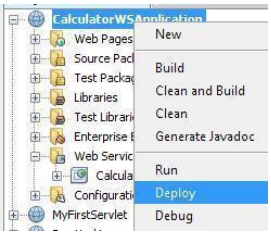
```
        return k;
    }
```

3) **Deploying and Testing the Web Service**

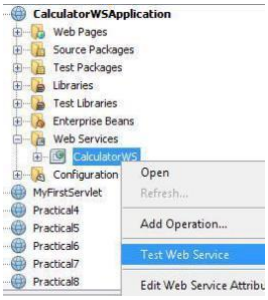
After you deploy a web service to a server, you can use the IDE to open the server's test client, if the server has a test client. The GlassFish and WebLogic servers provide test clients.

A. **To test successful deployment to a GlassFish or WebLogic server:**

1. Right-click the project and choose Deploy. The IDE starts the application server, builds the application, and deploys the application to the server



2. In the IDE's Projects tab, expand the Web Services node of the CalculatorWSApplication project. Right-click the CalculatorWS node, and choose Test Web Service.



3. The IDE opens the tester page in your browser, if you deployed a web application to the GlassFish server.
4. If you deployed to the GlassFish server, type two numbers in the tester page, as shown below:

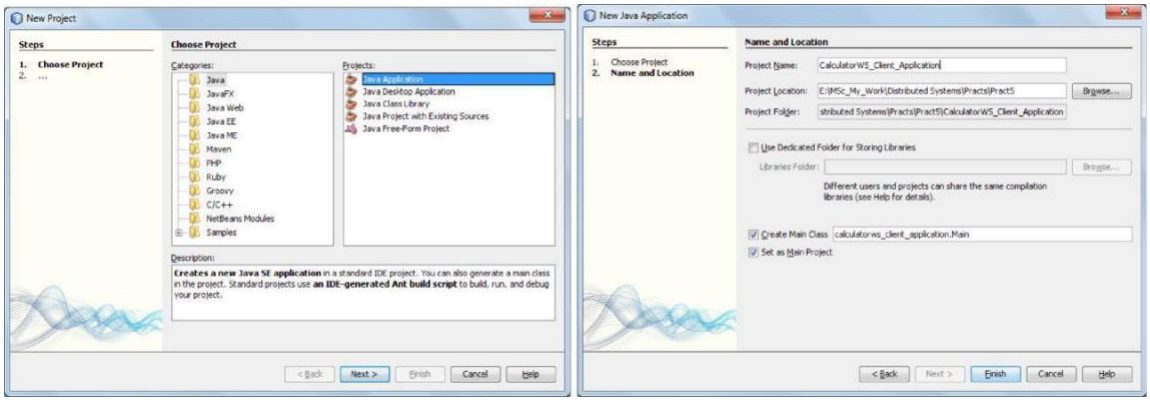
5. The sum of the two numbers is displayed:

4) **Consuming the Web Service**

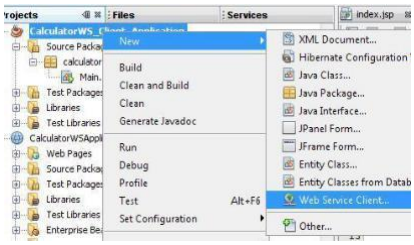
Now that you have deployed the web service, you need to create a client to make use of the web service's add method.

1. **Client: Java Class in Java SE Application**

1. Choose File > New Project. Select Java Application from the Java category. Name the project CalculatorWS_Client_Application. Leave Create Main Class selected and accept all other default settings. Click Finish.

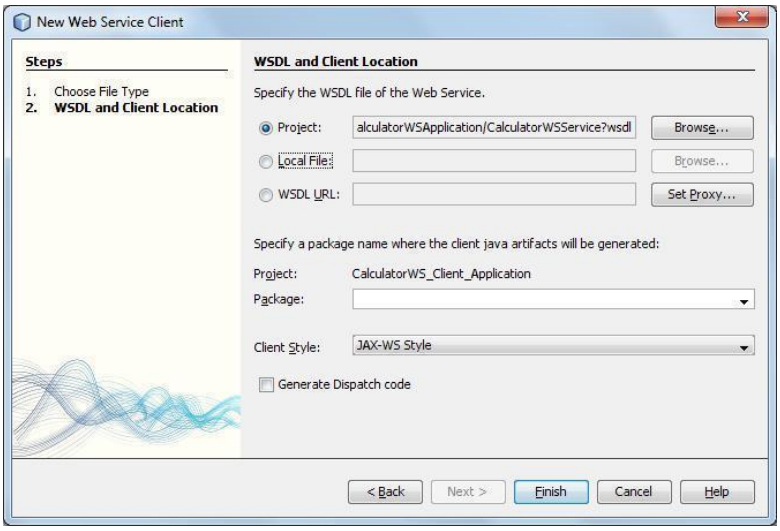


2. Right-click the CalculatorWS_Client_Application node and choose New > Web Service Client. The New Web Service Client wizard opens.



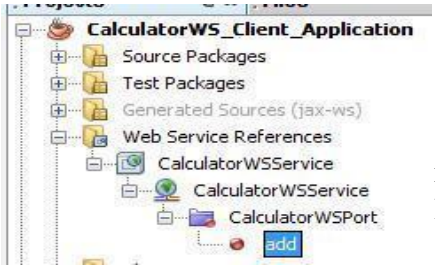
3. Select Project as the WSDL source. Click Browse. Browse to the CalculatorWS web service in the CalculatorWSApplication project. When you have selected the web service, click OK.

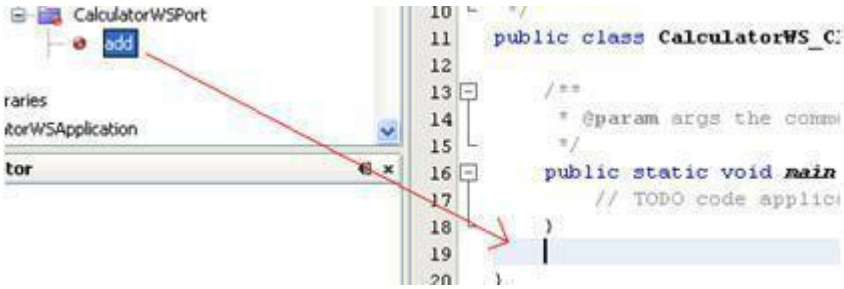
4. Do not select a package name. Leave this field empty.



5. Leave the other settings at default and click Finish. The Projects window displays the new web service client, with a node for the add method that you created:

6. Double-click your main class so that it opens in the Source Editor. Drag the add node below the main() method.





You now see the following:

```
public static void main(String[] args)
{
   // TODO code application logic here
}
private static int add(inti, int j)
{
        org.me.calculator.CalculatorWS_Service service = new
org.me.calculator.CalculatorWS_Service();
        org.me.calculator.CalculatorWS port = service.getCalculatorWSPort();
        return port.add(i, j);
}
```

7. In the main() method body, replace the TODO comment with code that initializes values for i and j, calls add(), and prints the result.

```
public static void main(String[] args)
{
   inti = 3;
   int j = 4;
   int result = add(i, j);
   System.out.println("Result = " + result);
}
```

8. Surround the main() method code with a try/catch block that prints an exception.

```
public static void main(String[] args)
{
   try
  {
      inti = 3;

      int j = 4;
      int result = add(i, j);
      System.out.println("Result = " + result);
   } catch (Exception ex) {
      System.out.println("Exception: " + ex);
   }
}
```

9. Right-click the project node and choose Run.

The Output window now shows the sum:

```
compile:
run:
Result = 7
BUILD SUCCESSFUL (total time: 1 second)
```

Practical 5B: Implementing Web Service that connects to MySQL database.

Building Web Service:-

- JAX-WS is an important part of the Java EE platform.
- JAX-WS simplifies the task of developing Web services using Java technology.
- It provides support for multiple protocols such as SOAP, XML and by providing a facility for supporting additional protocols along with HTTP.
- With its support for annotations, JAX-WS simplifies Web service development and reduces the size of runtime files.
- Here basic demonstration of using IDE to develop a JAX-WS Web Service is given.
- After creating the web service, create web service clients that use the Web service over a network which is called consuming a web service.

- The client is a servlet in a web application.
- Let's build a Web Service that returns the book name along with its cost for a particular ISBN.
- To begin building this service, create the data store. The server will access the data stored in a MySQL table to serve the client.

1)

2) **Creating MySQL DB Table**

```
create database bookshop;
use bookshop;
```

Create a table named Books that will store valid books information

```
create table books(isbn varchar(20) primary key, bookname varchar(100), bookprice
varchar(10));
```

Insert valid records in the Books table

```
insert into books values("111-222-333","Learn My SQL","250");
insert into books values("111-222-444","Java EE 6 for Beginners","850");
insert into books values("111-222-555","Programming with Android","500");
insert into books values("111-222-666","Oracle Database for you","400");
insert into books values("111-222-777","Asp.Net for advanced programmers","1250");
```

2) **Creating a web service**

i. **Choosing a container**

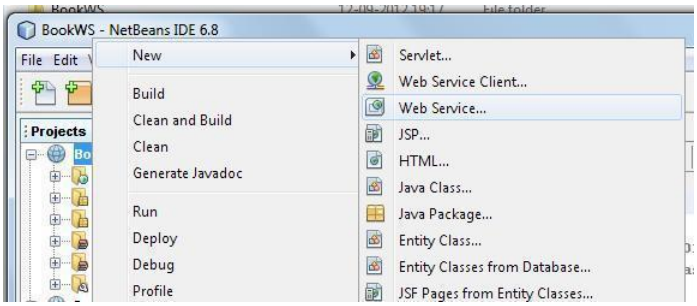
- Web service can be either deployed in a Web container or in an EJB container.
- If a Java EE 6 application is created, use a Web container because EJBs can be placed directly in a Web application.

ii. **Creating a web application**

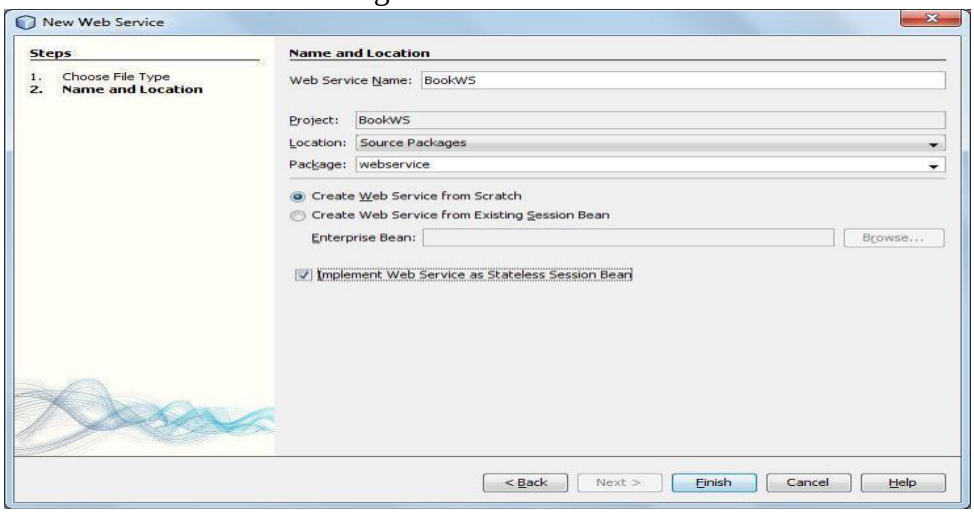
- To create a Web application, select File - New Project.
- New Project dialog box appears. Select Java Web available under the Categories section and Web Application available under the Projects section. Click Next.
- New Web Application dialog box appears. Enter BookWS as the project name in the Project Name textbox and select the option Use Dedicated Folder for Storing Libraries.
- Click Next. Server and Settings section of the New Web Application dialog box appears. Choose the default i.e. GlassFish v3 Domain as the Web server, the Java EE 6 Web as the Java EE version and the Context Path.
- Click –Finish
- The Web application named BookWS is created.

iii. **Creating a web service**

- Right-click the BookWS project and select New -> Web Service as shown in diagram.



- New Web Service dialog box appears. Enter the name BookWS in the Web Service Name textbox, webservice in the Package textbox, select the option Create Web Service from scratch and also select the option implement web service as a stateless session bean as shown in the diagram.



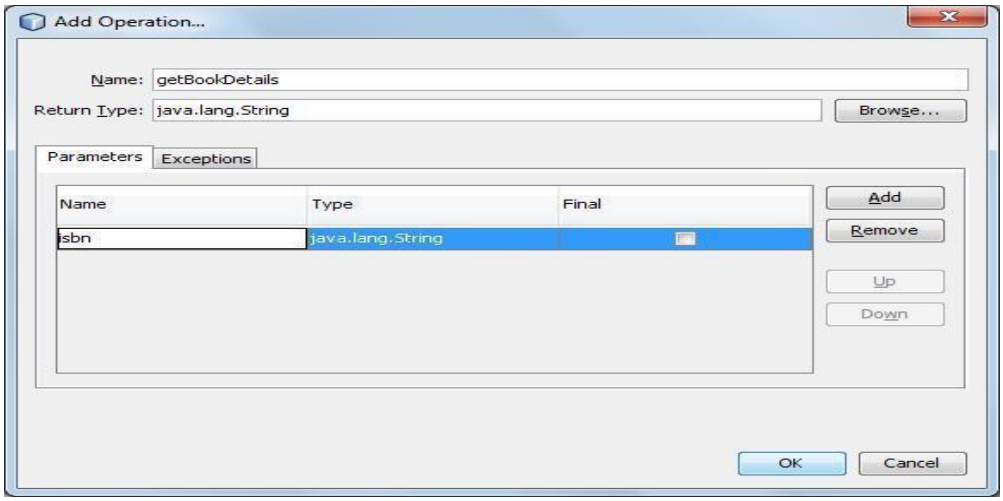
- Click Finish.
- The web service in the form of java class is ready.

3) **Designing the web service**

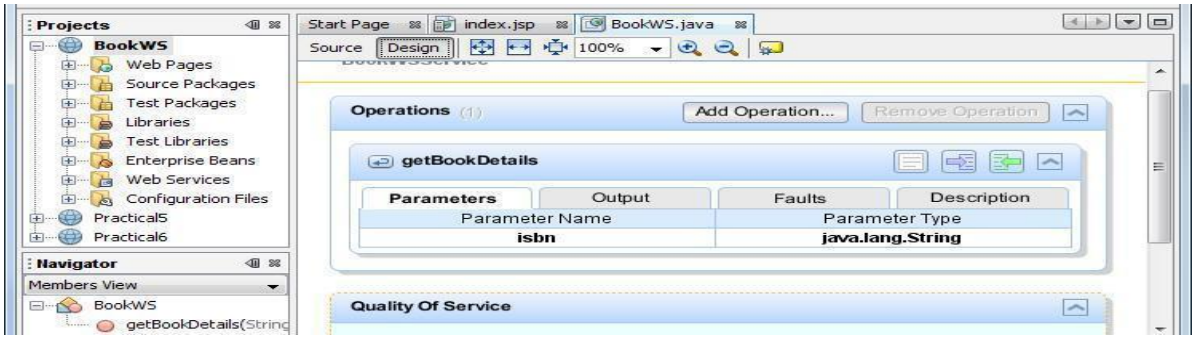
Now add an operation which will accept the ISBN number from the client to the web service.

i. **Adding an operation to the web service**

- Change the source view of the BookWS.java to design view by clicking Design available just below the name of the BookWS.java tab.
- The window changes as shown in the diagram.
- Click Add Operation available in the design view of the web service.
- Add Operation dialog appears. Enter the name getBookDetails in the Name textbox and java.lang.String in the Return Type textbox as shown in the diagram.
- In Add Operation dialog box, click Add and create a parameter of the type String named isbn as shown in the diagram.

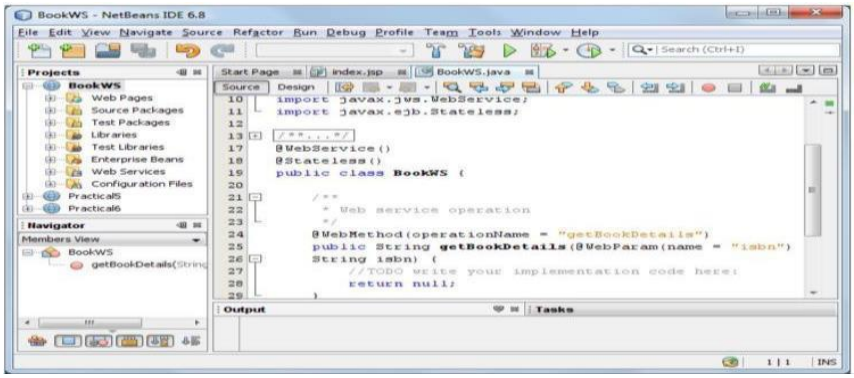


- Click Ok. The design view displays the operation added as shown in the diagram.



- Click Source. The code spec expands due to the operation added to the web service as shown in the diagram.

- Modify the code spec of the web service BookWS.java.



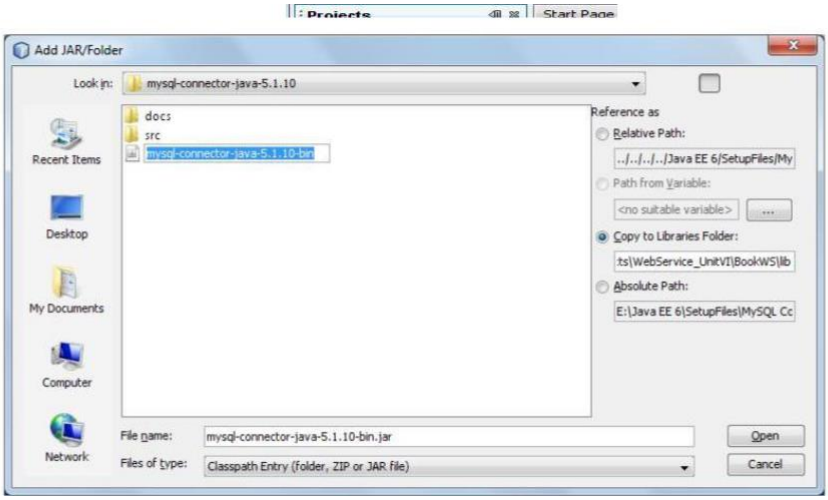
Code Spec

```
packagewebservice;
importjava.sql.*;
importjavax.jws.WebMethod;
importjavax.jws.WebParam;
```

```
importjavax.jws.WebService;
importjavax.ejb.Stateless;
@WebService()
@Stateless()
public class BookWS {
   /**
    * Web service operation
    */
   @WebMethod(operationName = "getBookDetails") public
   String getBookDetails(@WebParam(name = "isbn")
   String isbn) {
      //TODO write your implementation code here
      Connection dbcon = null;
      Statement stmt = null;
      ResultSetrs = null;
      String query = null;
      try
      {
         Class.forName("com.mysql.jdbc.Driver").newInstance();
         dbcon =
DriverManager.getConnection("jdbc:mysql://localhost/bookshop","root","123");
         stmt = dbcon.createStatement();
         query = "select * from books where isbn = '" +isbn+
         "'"; rs = stmt.executeQuery(query); rs.next();
         String bookDetails = "<h1>The name of the book is <b>"
+rs.getString("bookname") + "</b> and its cost is <b>" +rs.getString("bookprice") +
"</b></h1>.";
         returnbookDetails;
      }
      catch(Exception e)
      {
         System.out.println("Sorry failed to connect to the database.." + e.getMessage());
      }
      return null;
   }
```

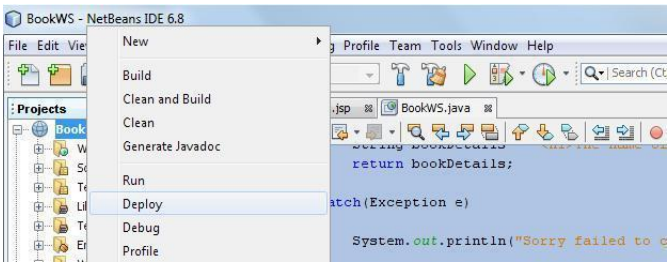
4) **Adding the MySQL connector**

- We need to add a reference of MySQL connector to our web service. It is via this connector that our web service will be able to communicate with the database.
- Right click on the libraries and select Add JAR/Folder as shown in the diagram.
- Choose the location where mysql-coonector-java-5.1.10-bin is located, select it and click on open as shown.



5) **Deploying and testing the web service**

- When a web service is deployed to a web container, the IDE allows testing the web service to see if it functions as expected.
- The tester application provided by GlassFish, is integrated into the IDE for this purpose as it allows the developer to enter values and test them.
- No facility for testing whether an EJB module is deployed successfully is currently available.
- To test the BookWS application, right click the BookWS project and select Deploy as shown in the diagram.



- The IDE starts the server, builds the application and deploys the application to the server.
- Follow the progress of these operations in the BookWS (run-deploy) and GlassFish v3 Domain tabs in the Output view.
- Now expand the web services directory of the BookWS project, right-click the BookWS Web service and select Test web service as shown in the diagram.
- The IDE opens the tester page in the web browser, if the web application is deployed using GlassFish server as shown in the figure.

- Enter the ISBN number as shown in the diagram.
- Click getBookDetails. The book name and its cost are displayed as shown in the diagram.

6) **Consuming the web service**

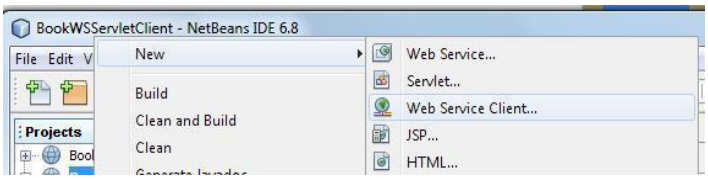
- Once the web service is deployed, the next most logical step is to create a client to make use of the web service's getBookDetails() method.

i. **Creating a web application**

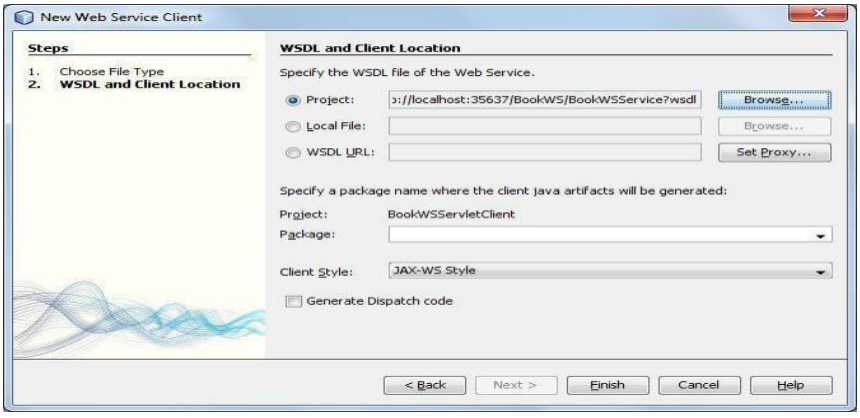
- To create a web application, select File -> New Project.
- New project dialog box appears, select java web available under the categories section and web application available under the projects section. Click Finish.
- New web application dialog box appears. Enter BookWSServletClient as the project name in the Project Name textbox and select the option Use Dedicated Folder for Storing Libraries.
- Click Next. Server and settings section of the new web application, dialog box appears. Choose the default i.e. GlassFish v3 Domain as the web serevr, the Java EE 6 web as the Java EE version and the context path.
- Click Finish.
- The web application named BookWSServletClient is created.

ii. **Adding the web service to the client application**

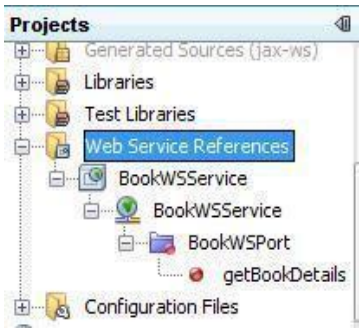
- Right-click the BookWSServletClient project and select New -> Web Service Client as shown in the diagram.



- New Web Service Client dialog box appears. In the Project section, click Browse and browse through the web service which needs to be consumed. Click ok. The name of the web service appears in the New Web Service Client as shown in the diagram.



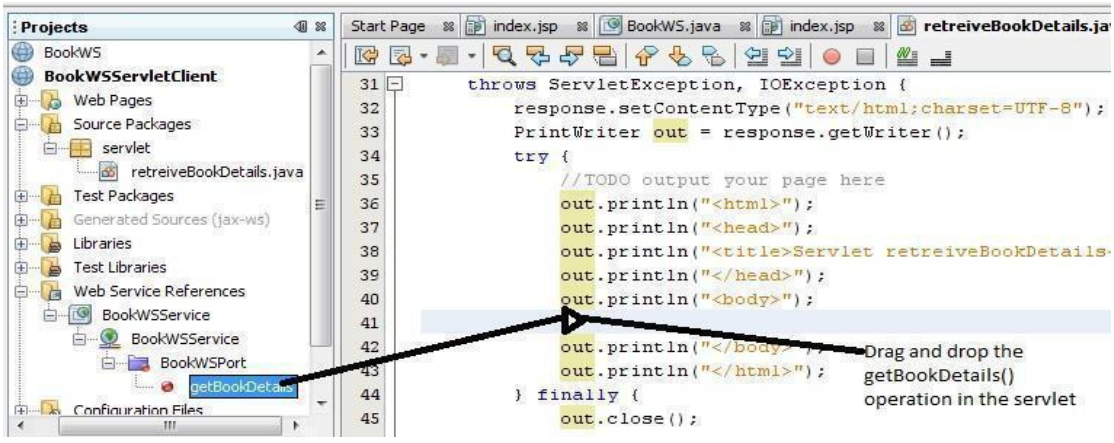
- Leave the other settings as it is. Click Finish
- The Web Service Reference directory is added to the BookWSServletClient application as shown in the diagram. It displays the structure of the newly created client including the getBookDetails() method created earlier.



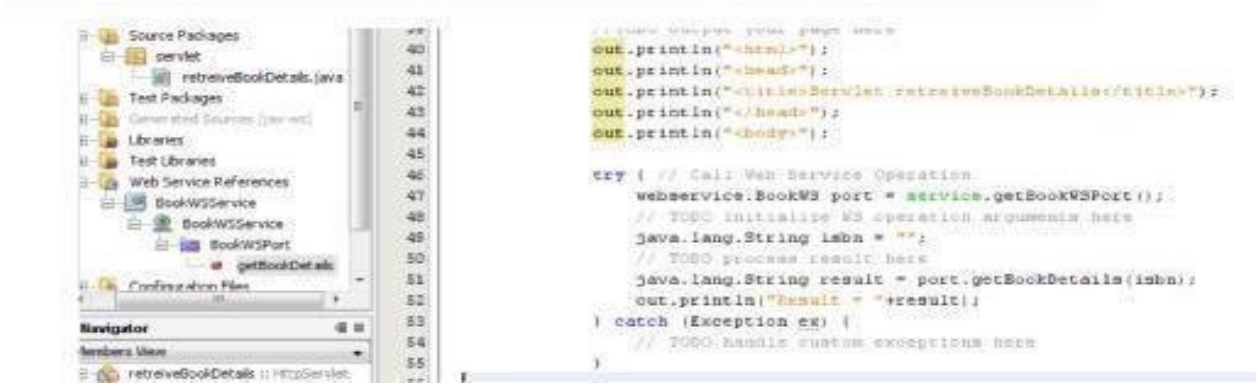
iii. Creating a servlet

- Create retreiveBookDetails.java using NetBeans IDE.
- Right click source package directory, select New -> Servlet.
- New Servlet dialog box appears. Enter retreiveBookDetails in the Class Name textbox and enter servlet in the package textbox.
- Click Next. Configure Servlet Deployment section of the New Servlet dialog box appears. Keep the defaults.
- Click Finish.
- This creates the servlet named retreiveBookDetails.java in the servlet package.
- retreiveBookDetails.java is available with the default skeleton created by the NetBeans IDE which needs to be modified to hold the application logic.
- In the retreieveBookDetails.java source code, remove the following comments available in the body of the processRequest() method.

  /*TODO output your page here*/
  Replace the following code spec:
  out.println("<h1>Servlet retreiveBookDetails at " + request.getContextPath () + "</h1>");
  With the code spec of the getBookDetails() operation of the web service by dragging and dropping the getBookDetails operation as shown in the diagram.



- The Servlet code spec changes as shown in the diagram



- The web service is instantiated by the @WebServiceRef annotation.
- Now change the following code spec:
  java.lang.Stringisbn = “”;
  to
  java.lang.Stringisbn = request.getParameter(“isbn”);

iv. Creating an HTML form

- Once the web service is added and the servlet is created, the form to accept ISBN from the user needs to be coded.
- Since NetBeans IDE by default [as a part of Web Application creation] makes available index.jsp file. Modify it to hold the following code spec.

```
<%@page contentType="text/html" pageEncoding="UTF-
8"%><!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-
    8"><title>SOAP Cleint - Get Book Details</title>
  </head>
  <body bgcolor="pink">
    <form name="frmgetBookDetails" method="post"
      action="retreiveBookDetails"><h1>
        ISBN : <input type="text"
      name="isbn"/><br><br></h1>

      <input type="submit"
    value="Submit"/></form>
  </body>
</html>
```

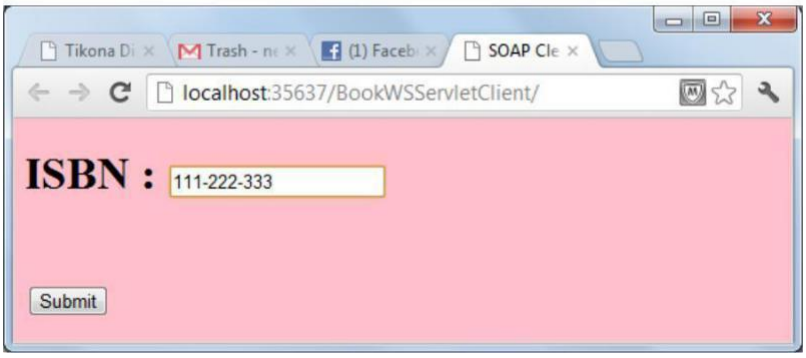
v. Building the Web Application

- Build the web application.

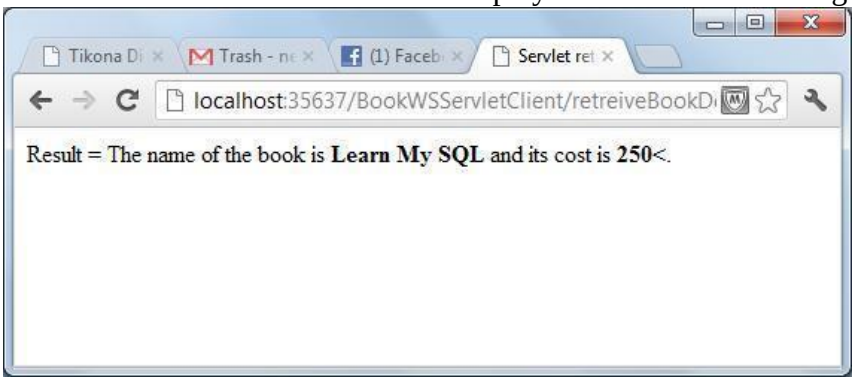
- Right click BookWSServletClient project and select Build.
- Once the Build menu item is clicked the details about the compilation and building of the BookWSServletClient Web application appears in the output – BookWSServletClient (dist) window.

vi. **Running the Application**

- Once the compilation and building of the web application is done run the application.
- Right click the BookWSServerCleint project and select run.
- Once the run processing completes in NetBeans IDE a web browser is automatically launched and the BookWSServletCleint application is executed as shown in the diagram.
- Enter the ISBN as shown in the diagram



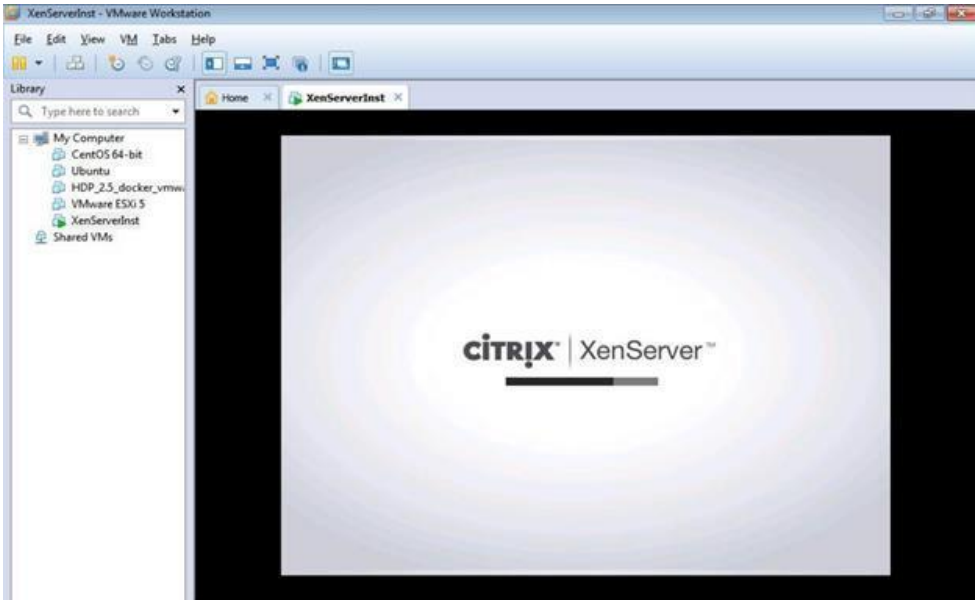
- Click Submit. The book name and its cost are displayed as shown in the diagram.



Practical:06

Aim: Implement Xen virtualization and manage with Xen Center

Install **XenServer** in VMware Workstation and select Guest operating system as Linux.



Note IP Address – " 192.168.124.137" ping it from command prompt.

Now Install Citrix App if not installed

Now Open Citrix XenCenter – and Click and **Add Server.**

Fill IP address copied from Installation and User name as "root" and Password as "root123" which we had given during installation and Click on Add.

Then click on Ok

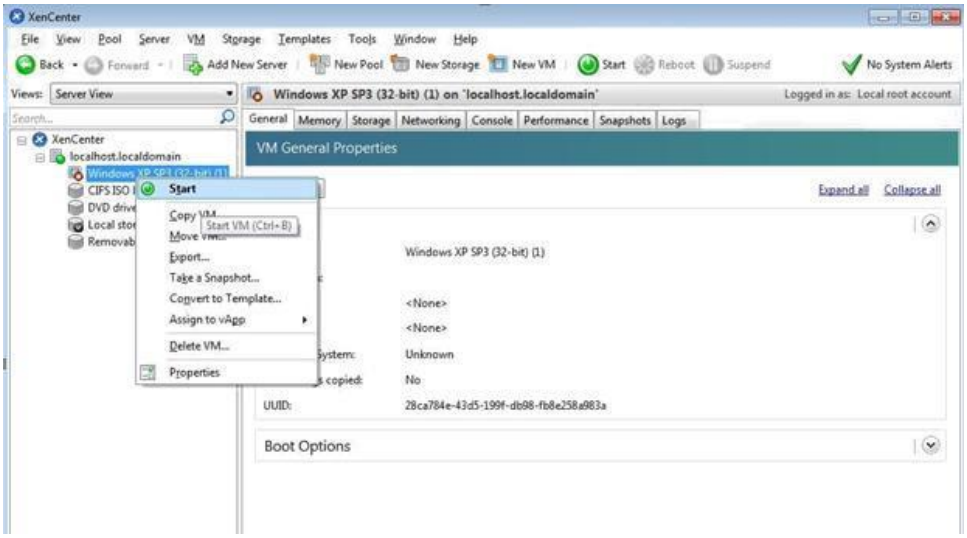
Now Click on New Storage

Select Window File Sharing (CIFS) and click on next Uncheck Auto generate option Click on Next.

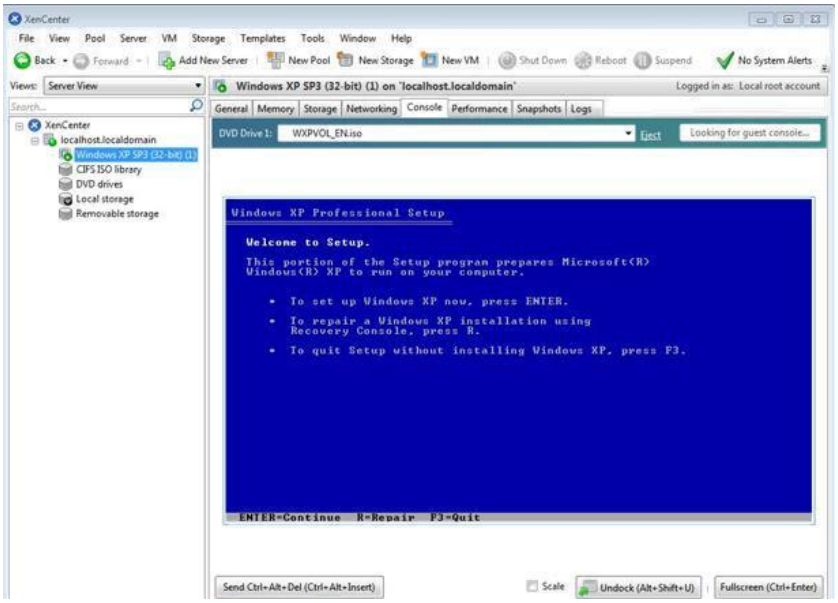
Provide the path of shared windows XP image and enter local pc credential , click on Finish

Click on New VM – and Windows XP SP3

Select ISO file and click on next –
Click on Next –
Uncheck – Start the new VM and click on create now Now Right click on Windows XP and Start -



Installation is successful and virtual node has been created if we get below Welcome screen of Windows XP machine.

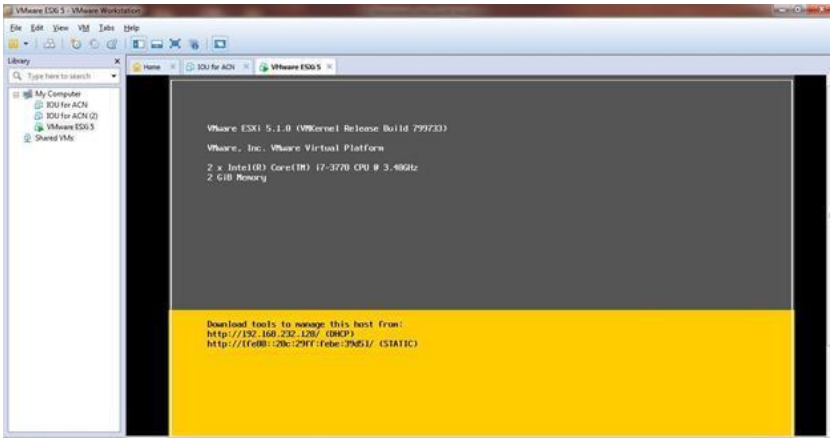


Practical: 07

1. **Aim:**Implement virtualization using VMWare ESXi Server and managing with vCenter

Steps:

Install **ESXiiso**in VMWare workstation.



InstallVMware vSphere Client

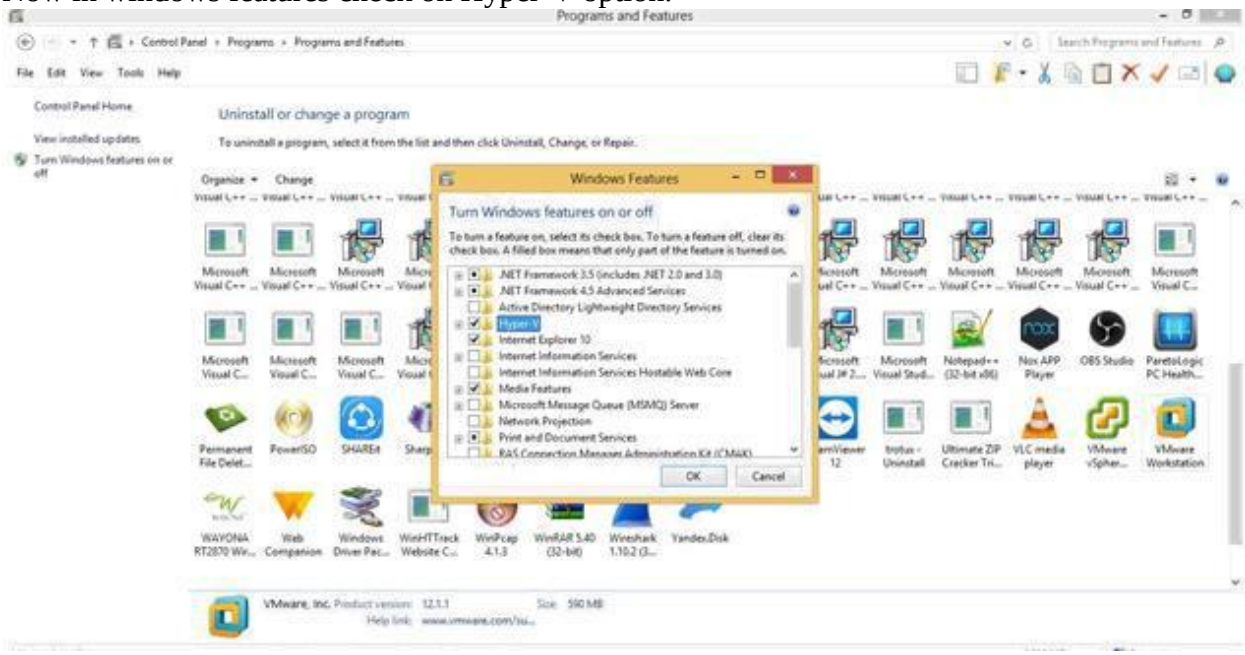
In vSphere create new **Virtual Machine.** Install Windows XP iso file and open it.

Practical: 08

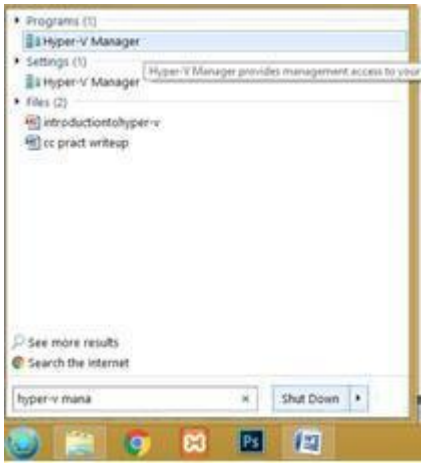
Aim: Implement Windows Hyper V virtualization

First we have to uninstall vmware software if already installed on computer becausethe VMware Workstation installer does not support running on a Hyper-V virtual machine. after uninstalling vmware we can proceed to next step go to controlpanel and click on uninstall a program.

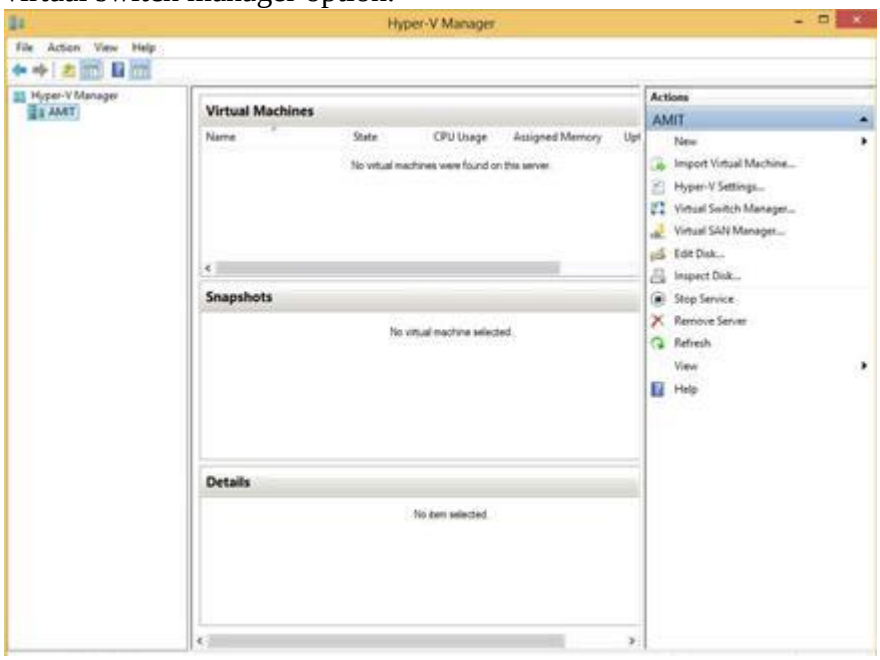
Click on Turn windows features on or off.
Now in windows features check on Hyper-V option.



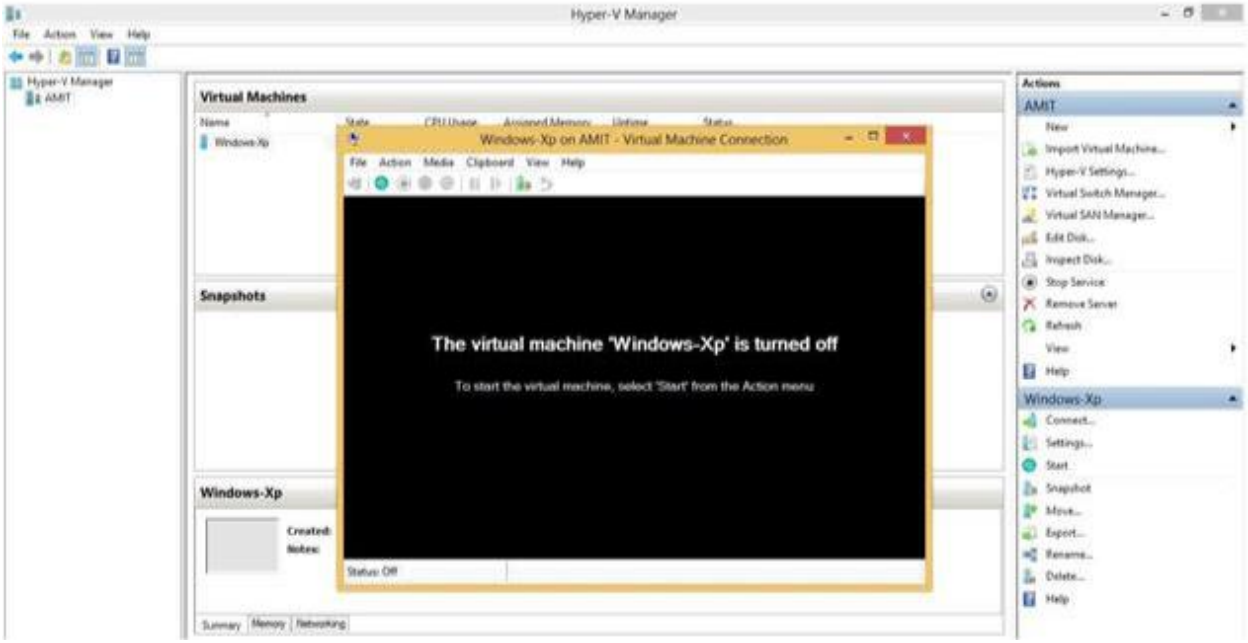
After Restart **Search for hyper-v manager in search box and click on that.**



for creating virtual machine first we have to create virtual switch click on virtual switch manager option.



Select External as a connection type and then click on create virtual switch.
Create new Virtual switch and install windows XP .iso



and virtual machine will start.

Practical: 09

Aim: Develop application for Microsoft Azure.

Step 1:

To develop an application for Windows Azure on Visual Studio install the **"Microsoft Azure SDK for .NET (VS 2010) – 2.8.2.1"**

Step2:

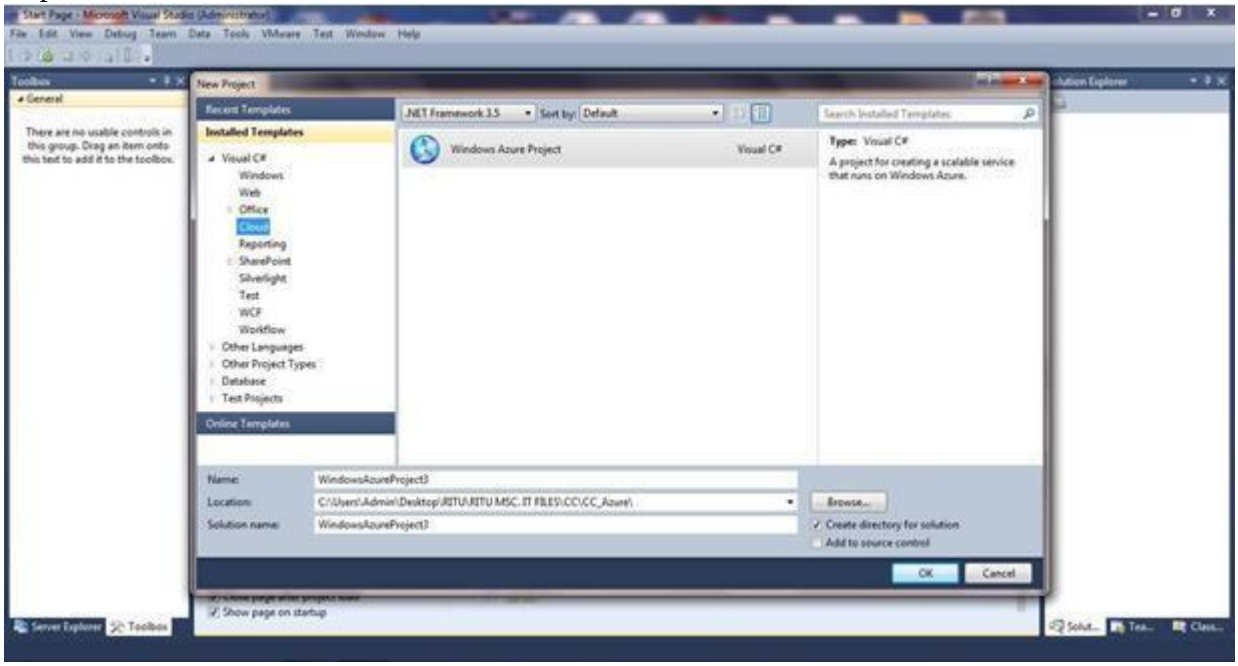
Turn windows Features ON or OFF:

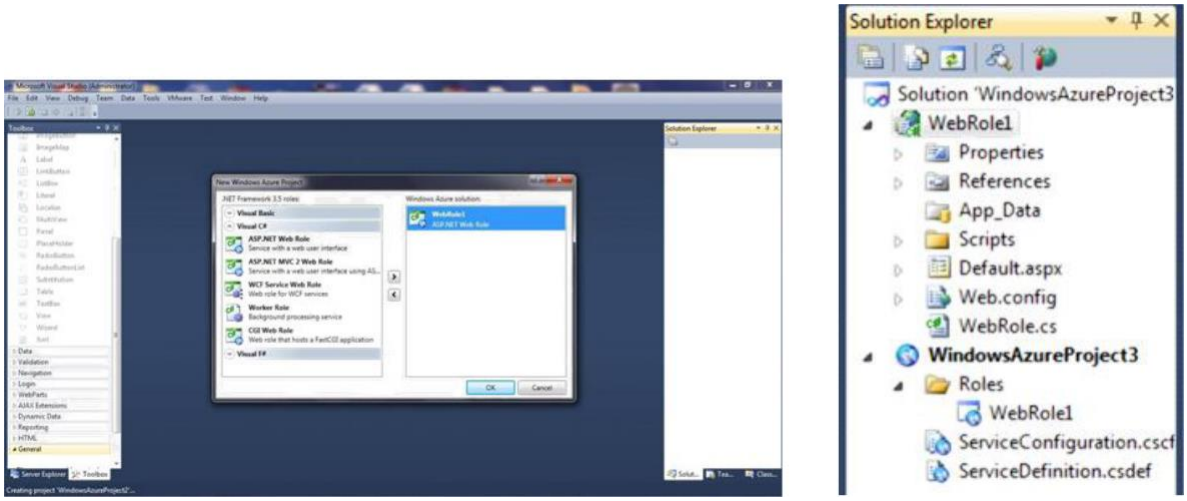
Go to Control panel and click on programs.
Turn Windows features on or off.

Step3:

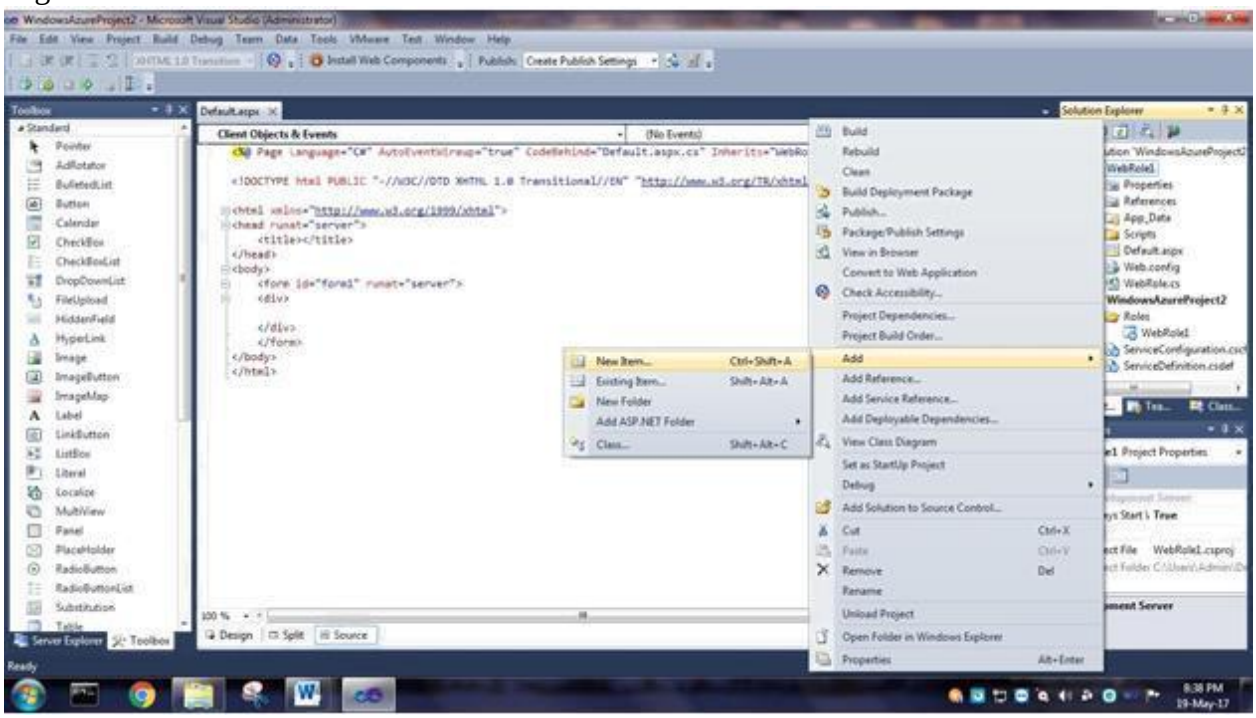
Now, Start the visual studio 2010 and Go To File->New->Project

Expand Visual C#-> Select Cloud

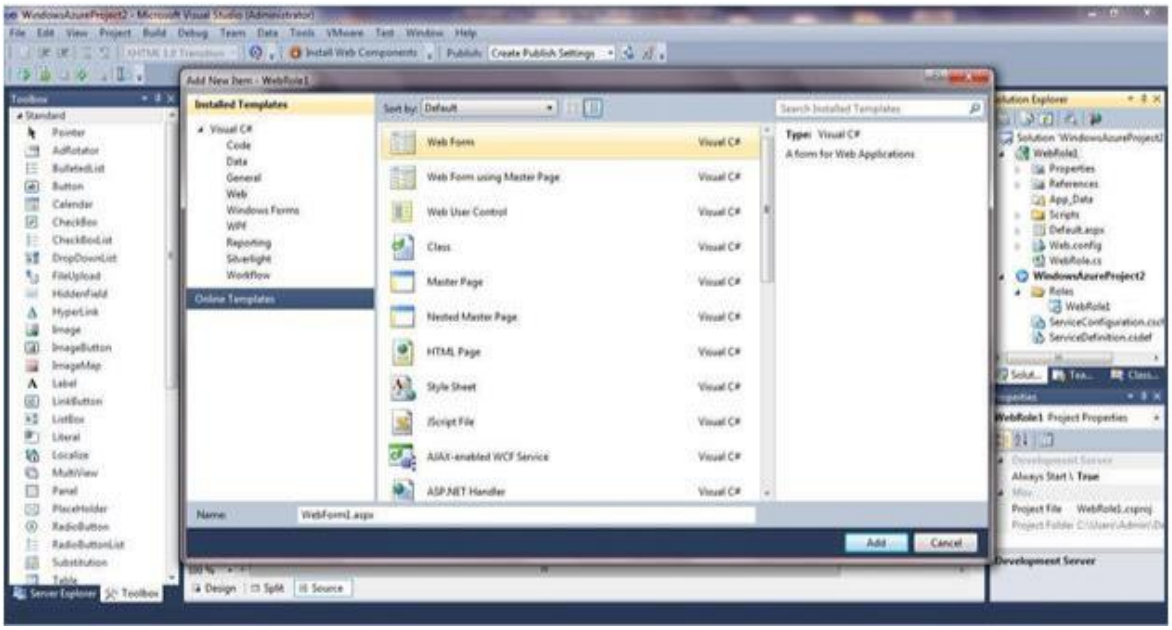




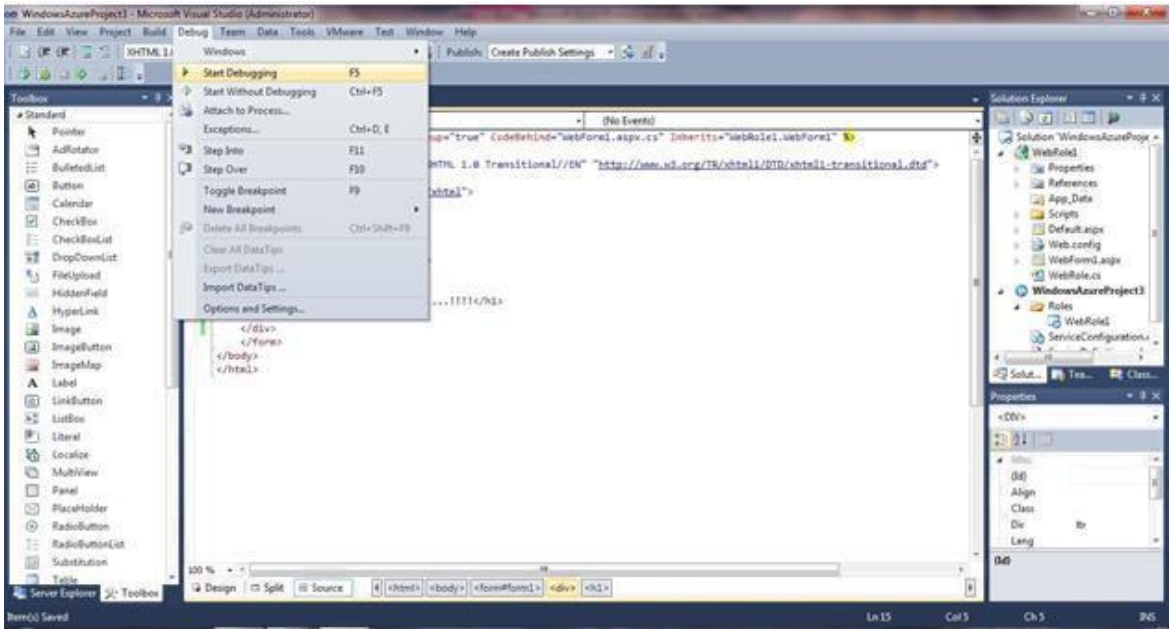
Right Click on WebRole1>>ADD>>New Item



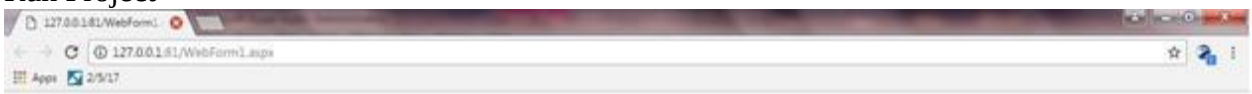
Add a New web Form. Give it a name. Click Add



Deploy the project:



Run Project



Welcome TO Windows Azure.....!!!!

Practical: 10

Aim: Develop application for Google App Engine

Open Eclipse Luna. Go to **Help Menu Install New Software…**

In **Install** window Click on the **"Add"** button besides the **Work with** textbox. **Add Repository** window appears. Enter the **Location** as

"https://dl.google.com/eclipse/plugin/4.4" and click on "**OK**" button. From the available softwares select the required softwares and tools as shown in the below image for the **GAE**. Then click on the **"Next"** button.

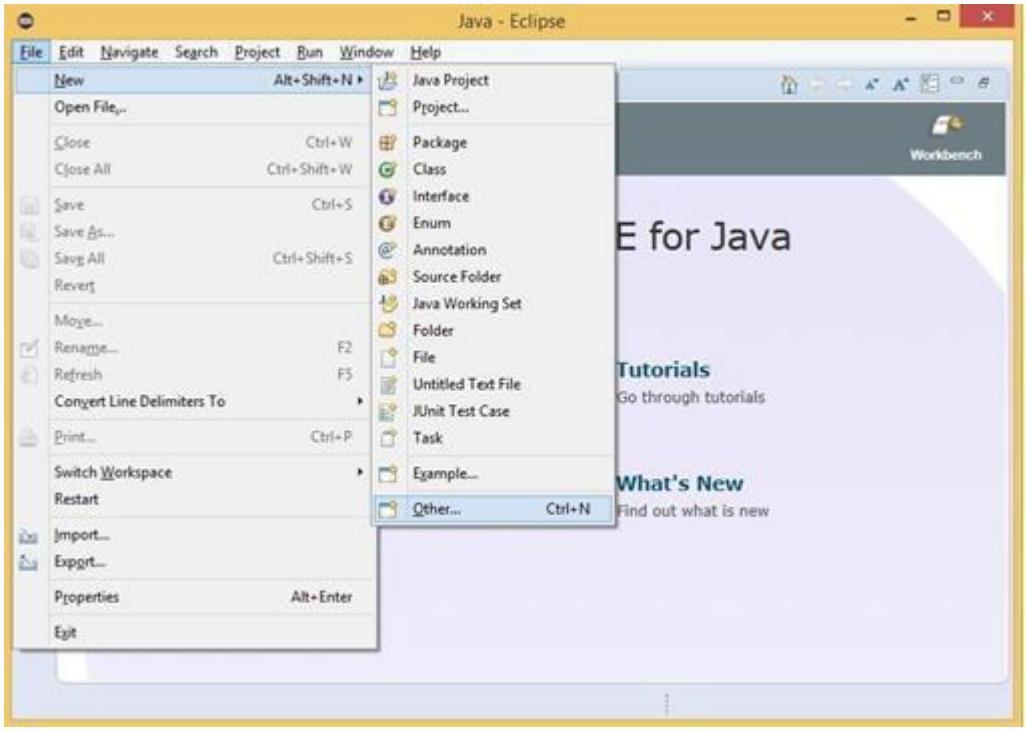
In the **Install Details** window click on **"Next"** button.

In the Next Window "Review the Items to be Installed" then click on **"Next"**

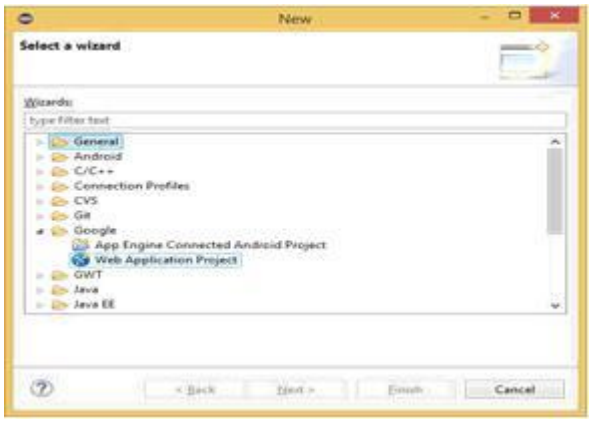
In the next window for Review Licenses select the option **"I accept……"** and click on **"Finish"** button.

After Installation you will get option to **"Restart Eclipse",** click on **Yes.** So that the software you selected gets updated...

Now, go to **File Menu_New_Other.**

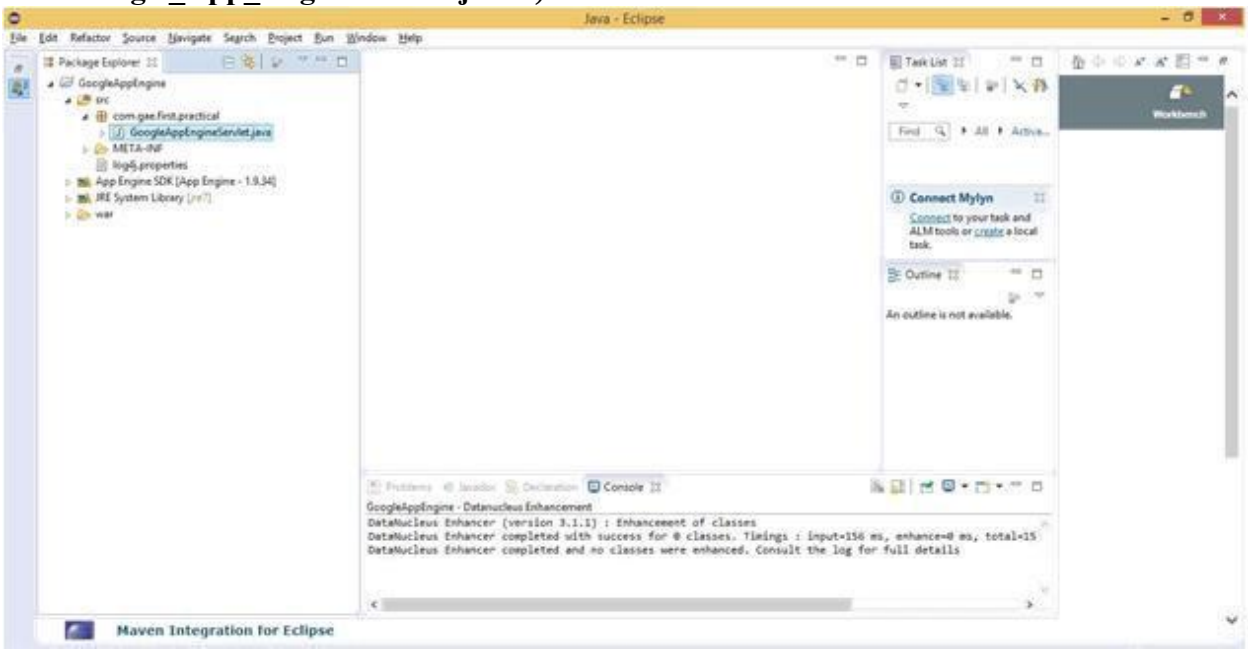


In the New window select **Google_Web Application Project** and click on **"Next"** button.

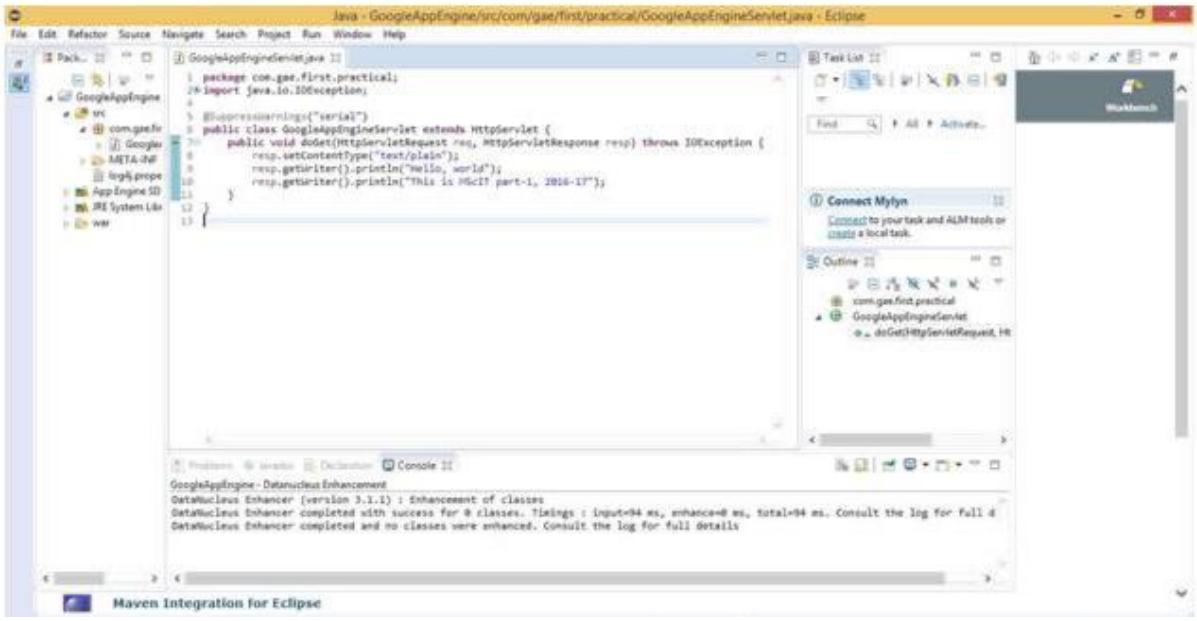


Enter the details for the new Web application project. Deselect the **Use Google Web Toolkit** option under the section **Google SDKs**. Click on the **"Finish"** button.

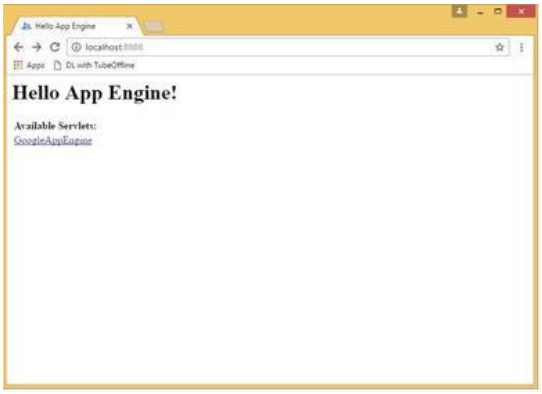
From the **Package Explorer** open the **.java** file (Here it is **"Google_App_EngineServlet.java"**).



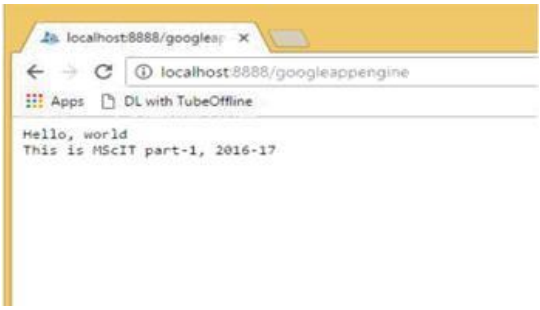
Edit the file as required (Unedited file too can be used. Here the editing is done to "what should be displayed" on the browser). **Save** the file. Click on the **Run** option available on the Tools bar.



In the browser (Here, Google Chrome) type the address as **"localhost:8888"which is "Default"**.



In **localhost:8888** the link to the **Google_App_EngineServlet.java** file as **Google_App_Engine**is displayed. Click on this link. It will direct you to **"localhost:8888/Google_App_Engine"**.



The **output text entered** in the **java** program is **displayed as the output** when clicked the link
"Google_App_Engine"

SOFT COMPUTING TECHNIQUES [PSIT1P4]

S K SOMAIYA COLLEGE OF ARTS, SCIENCE & COMMERCE
"Aurobindo"
Vidyavihar(East) Mumbai 400077

CERTIFICATE

This is to Certify that,
Mr./Ms. _____
Student of **M.Sc. I.T Part-I (Sem-I)** with seat no _____ and college enrolled roll no _____ has satisfactorily completed the practical in Information Technology Laboratory for the course **(PSIT104) Soft Computing Techniques** in the program of **INFORMATION TECHNOLOGY** from the **UNIVERSITY OF MUMBAI** for the academic year 2019-2020.

(Subject In-Charge) **(HOD)**

(Examiners)

_____

TABLE OF CONTENTS

7.	Implement the Following	
a	Write a program for Linear separation.	
b	Write a program for Hopfield network model for associative memory	
8.	Implement the Following	
a	Membership and Identity Operators | in, not in,	
b.	Membership and Identity Operatorsis, is not	
9.	Implement the Following	
a	Find ratios using fuzzy logic	
b	Solve Tipping problem using fuzzy logic	
10.	Implement the Following	
a	Implementation of Simple genetic algorithm	
b	Create two classes: City and Fitness using Genetic algorithm	

Practical 1

Practical 1 a: Design a simple linear neural network model.

```
x=float(input("Enter value of x:"))
w=float(input("Enter value of weight w:"))
b=float(input("Enter value of bias b:"))
net = int(w*x+b)
if(net<0):
   out=0
elif((net>=0)&(net<=1)):
   out =net
else:
   out=1
print("net=",net)
print("output=",out)
```

1b: Calculate the output of neural net using both binary and bipolar sigmoidal function.

For the network shown in the figure 1, calculate the net input to output neuron.

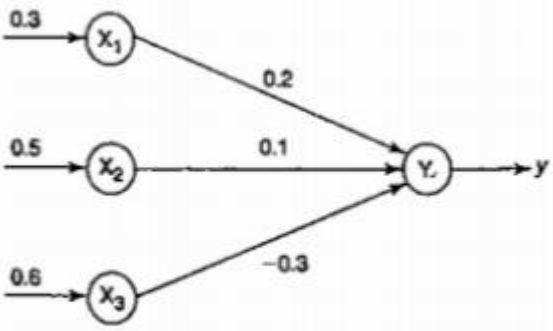


Figure 1 Neural net.

Code :

```
# number of elements as input
n = int(input("Enter number of elements : "))

# In[2]:
print("Enter the inputs")
inputs = [] # creating an empty list for inputs

# iterating till the range
for i in range(0, n):
        ele = float(input())
        inputs.append(ele) # adding the element

print(inputs)

# In[3]:
print("Enter the weights")
```

```
# creating an empty list for weights
weights = []

# iterating till the range
for i in range(0, n):
        ele = float(input())
        weights.append(ele) # adding the element

print(weights)

# In[4]:
print("The net input can be calculated as   Yin = x1w1 + x2w2 + x3w3")

# In[5]:
Yin = []
for i in range(0, n):
        Yin.append(inputs[i]*weights[i])
print(round(sum(Yin),3))
```

Output :

```
Enter number of elements : 3
Enter the inputs
0.3
0.5
0.6
[0.3, 0.5, 0.6]
Enter the weights
0.2
0.1
-0.3
[0.2, 0.1, -0.3]
The net input can be calculated as   Yin = x1w1 + x2w2 + x3w3
-0.07
```

1. **Problem statement :**

1. Calculate the net input for the network shown in Figure 2 with bias included in the network.

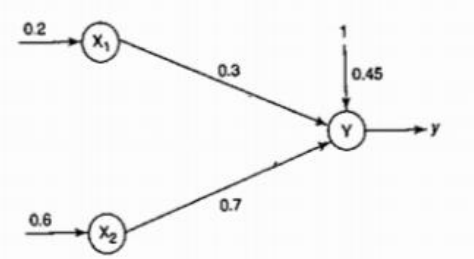


Figure 2 Simple neural net.

Code :

```
n = int(input("Enter number of elements : "))
print("Enter the inputs:")
inputs = [] # creating an empty list for inputs
for i in range(0, n):
        ele = float(input())
        inputs.append(ele) # adding the element
print(inputs)
print("Enter the weights:")
weights = []
for i in range(0, n):
        ele = float(input())
        weights.append(ele) # adding the element
print(weights)
b=float(input("Enter bias value:"))
print("The net input can be calculated as   Yin = b + x1w1 + x2w2:")
Yin = []
for i in range(0, n):
        Yin.append(inputs[i]*weights[i])
print(round((sum(Yin)+b),3))
```

```
Enter number of elements : 2
Enter the inputs:
0.2
0.6
[0.2, 0.6]
Enter the weights:
0.3
0.7
[0.3, 0.7]
Enter bias value:0.45
The net input can be calculated as   Yin = b + x1w1 + x2w2:
0.93
```

Practical 2

Practical 2a: Implement AND/NOT function using McCulloch-Pits neuron (use binary data representation).

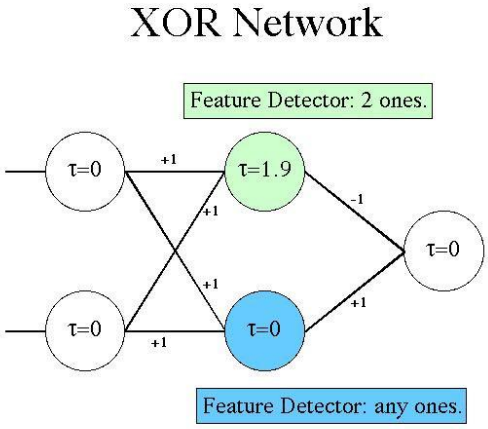
Code:

```
# enter the no of inputs
num_ip = int(input("Enter the number of inputs : "))
#Set the weights with value 1
w1 = 1
w2 = 1
print("For the ", num_ip , " inputs calculate the net input using yin = x1w1 + x2w2 ")
x1 = []
x2 = []
for j in range(0, num_ip):
    ele1 = int(input("x1 = "))
    ele2 = int(input("x2 = "))
    x1.append(ele1)
    x2.append(ele2)
print("x1 = ",x1)
print("x2 = ",x2)
n = x1 * w1
m = x2 * w2
Yin = []
for i in range(0, num_ip):
        Yin.append(n[i] + m[i])
print("Yin = ",Yin)
#Assume one weight as excitatory and the other as inhibitory, i.e.,
Yin = []
for i in range(0, num_ip):
        Yin.append(n[i] - m[i])
print("After assuming one weight as excitatory and the other as inhibitory Yin = ",Yin)
#From the calculated net inputs, now it is possible to fire the neuron for input (1, 0)
#only by fixing a threshold of 1, i.e., θ ≥ 1 for Y unit.
#Thus, w1 = 1, w2 = -1; θ ≥ 1
Y=[]
for i in range(0, num_ip):
    if(Yin[i]>=1):
        ele=  1
        Y.append(ele)
    if(Yin[i]<1):
        ele=  0
        Y.append(ele)
print("Y = ",Y)
```

Output:

```
Enter the number of inputs : 4
For the  4  inputs calculate the net input using yin = x1w1 + x2w2

x1 = 0

x2 = 0

x1 = 0

x2 = 1

x1 = 1

x2 = 0

x1 = 1

x2 = 1
x1 =  [0, 0, 1, 1]
x2 =  [0, 1, 0, 1]
Yin =  [0, 1, 1, 2]
 After assuming one weight as excitatory and the other as inhibitory Yin =  [0, 1, -1, 0]
Y =  [0, 1, 0, 0]

In [14]:
```

Practical 2b: Generate XOR function using McCulloch-Pitts neural net



The XOR (exclusive or) function is defined by the following truth table:

Input1	Input2	XOR Output
0	0	0
0	1	1
1	0	1
1	1	0

```
#Getting weights and threshold value
import numpy as np
print('Enter weights')
w11=int(input('Weight w11='))
w12=int(input('weight w12='))
w21=int(input('Weight w21='))
w22=int(input('weight w22='))
v1=int(input('weight v1='))
```

```
v2=int(input('weight v2='))
print('Enter Threshold Value')
theta=int(input('theta='))
x1=np.array([0, 0, 1, 1])
x2=np.array([0, 1, 0, 1])
z=np.array([0, 1, 1, 0])
con=1
y1=np.zeros((4,))
y2=np.zeros((4,))
y=np.zeros((4,))
while con==1:
    zin1=np.zeros((4,))
    zin2=np.zeros((4,))
    zin1=x1*w11+x2*w21
    zin2=x1*w21+x2*w22

    print("z1",zin1)
    print("z2",zin2)
    for i in range(0,4):
        if zin1[i]>=theta:
            y1[i]=1
        else:
            y1[i]=0

        if zin2[i]>=theta:
            y2[i]=1
        else:
            y2[i]=0

    yin=np.array([])
    yin=y1*v1+y2*v2
    for i in range(0,4):
        if yin[i]>=theta:
            y[i]=1
        else:
            y[i]=0

    print("yin",yin)
    print('Output of Net')
    y=y.astype(int)
    print("y",y)
    print("z",z)

    if np.array_equal(y,z):
        con=0
    else:
        print("Net is not learning enter another set of weights and Threshold value")
        w11=input("Weight w11=")
        w12=input("weight w12=")
```

```
    w21=input("Weight w21=")
    w22=input("weight w22=")
    v1=input("weight v1=")
    v2=input("weight v2=")
    theta=input("theta=")

print("McCulloch-Pitts Net for XOR function")
print("Weights of Neuron Z1")
print(w11)
print(w21)
print("weights of Neuron Z2")
print(w12)
print(w22)
print("weights of Neuron Y")
print(v1)
print(v2)
print("Threshold value")
print(theta)
```

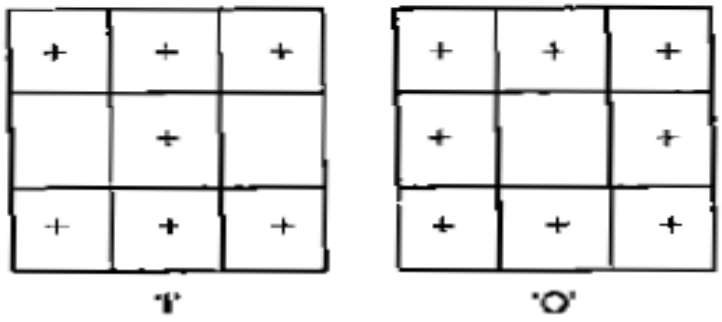
Output:

```
Enter weights
Weight w11=1
weight w12=-1
Weight w21=-1
weight w22=1
weight v1=1
weight v2=1
Enter Threshold Value
theta=1
z1 [ 0 -1  1  0]
z2 [ 0  1 -1  0]
yin [0. 1. 1. 0.]
Output of Net
y [0 1 1 0]
z [0 1 1 0]
McCulloch-Pitts Net for XOR function
Weights of Neuron Z1
1
-1
weights of Neuron Z2
-1
1
weights of Neuron Y
1
1
Threshold value
1
```

Practical 3

3a. Write a program to implement Hebb's rule.

Using the Hebb rule, find the weights required to perform the following classifications of the given input patterns shown in Figure 16. The pattern is shown as 3 × 3 matrix form in the squares. The "+" symbols represent the value "1" and empty squares indicate "−1." Consider "I" belongs to the members of class (so has target value 1) and "O" does not belong to the members of class (so has target value −1).




```
import numpy as np
#first pattern
x1=np.array([1,1,1,-1,1,-1,1,1,1])
#second pattern
x2=np.array([1,1,1,1,-1,1,1,1,1])
#initialize bais value
b=0
#define target
y=np.array([1,-1])

wtold=np.zeros((9,))
wtnew=np.zeros((9,))

wtnew=wtnew.astype(int)
wtold=wtold.astype(int)

bais=0
print("First input with target =1")
for i in range(0,9):
   wtold[i]=wtold[i]+x1[i]*y[0]
wtnew=wtold
b=b+y[0]
```

```
print("new wt =", wtnew)
print("Bias value",b)

print("Second input with target =-1")
for i in range(0,9):
   wtnew[i]=wtold[i]+x2[i]*y[1]
b=b+y[1]
print("new wt =", wtnew)
print("Bias value",b)
```

```
First input with target =1
new wt = [ 1  1  1 -1  1 -1  1  1  1]
Bias value 1
Second input with target =-1
new wt = [ 0  0  0 -2  2 -2  0  0  0]
Bias value 0
```

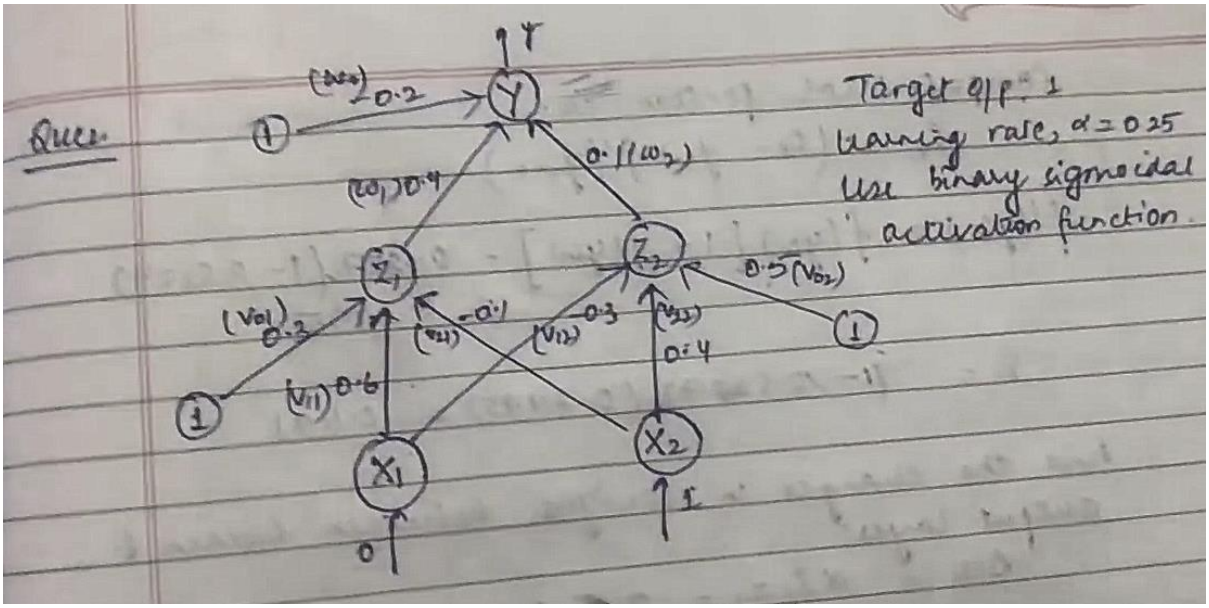
Practical 3b: Write a program to implement of delta rule.

```
#supervised learning
import numpy as np
import time
np.set_printoptions(precision=2)
x=np.zeros((3,))
weights=np.zeros((3,))
desired=np.zeros((3,))
actual=np.zeros((3,))

for i in range(0,3):
   x[i]=float(input("Initial inputs:"))

for i in range(0,3):
   weights[i]=float(input("Initial weights:"))

for i in range(0,3):
   desired[i]=float(input("Desired output:"))

a=float(input("Enter learning rate:"))

actual=x*weights
print("actual",actual)
print("desired",desired)

while True:
   if np.array_equal(desired,actual):
      break    #no change
   else:
      for i in range(0,3):
         weights[i]=weights[i]+a*(desired[i]-actual[i])

   actual=x*weights
   print("weights",weights)
   print("actual",actual)
   print("desired",desired)
print("*"*30)
print("Final output")
print("Corrected weights",weights)
print("actual",actual)
print("desired",desired)
```

```
Initial inputs:1
Initial inputs:1
Initial inputs:1
Initial weights:1
Initial weights:1
Initial weights:1
Desired output:2
Desired output:3
Desired output:4
Enter learning rate:1
actual [1. 1. 1.]
desired [2. 3. 4.]
weights [2. 3. 4.]
actual [2. 3. 4.]
desired [2. 3. 4.]
******************************
Final output
corrected weights [2. 3. 4.]
actual [2. 3. 4.]
desired [2. 3. 4.]
```

Practical 4

Practical 4a: Write a program for Back Propagation Algorithm




```
import numpy as np
import decimal
import math
np.set_printoptions(precision=2)
v1=np.array([0.6, 0.3])
v2=np.array([-0.1, 0.4])
w=np.array([-0.2,0.4,0.1])
b1=0.3
b2=0.5
x1=0
x2=1
alpha=0.25
print("calculate net input to z1 layer")
zin1=round(b1+ x1*v1[0]+x2*v2[0],4)
print("z1=",round(zin1,3))

print("calculate net input to z2 layer")
zin2=round(b2+ x1*v1[1]+x2*v2[1],4)
print("z2=",round(zin2,4))
print("Apply activation function to calculate output")
z1=1/(1+math.exp(-zin1))
z1=round(z1,4)
z2=1/(1+math.exp(-zin2))
z2=round(z2,4)
print("z1=",z1)
print("z2=",z2)
```

```
print("calculate net input to output layer")
yin=w[0]+z1*w[1]+z2*w[2]
print("yin=",yin)

print("calculate net output")
y=1/(1+math.exp(-yin))
print("y=",y)

fyin=y *(1- y)
dk=(1-y)*fyin
print("dk",dk)

dw1= alpha * dk * z1
dw2= alpha * dk * z2
dw0= alpha * dk

print("compute error portion in delta")

din1=dk* w[1]
din2=dk* w[2]
print("din1=",din1)
print("din2=",din2)

print("error in delta")
fzin1= z1 *(1-z1)
print("fzin1",fzin1)
d1=din1* fzin1
fzin2= z2 *(1-z2)
print("fzin2",fzin2)
d2=din2* fzin2

print("d1=",d1)
print("d2=",d2)

print("Changes in weights between input and hidden layer")
dv11=alpha * d1 * x1
print("dv11=",dv11)
dv21=alpha * d1 * x2
print("dv21=",dv21)
dv01=alpha * d1
print("dv01=",dv01)
dv12=alpha * d2 * x1
print("dv12=",dv12)
dv22=alpha * d2 * x2
print("dv22=",dv22)
dv02=alpha * d2
print("dv02=",dv02)
print("Final weights of network")
v1[0]=v1[0]+dv11
```

```
v1[1]=v1[1]+dv12
print("v=",v1)
v2[0]=v2[0]+dv21
v2[1]=v2[1]+dv22
print("v2",v2)
w[1]=w[1]+dw1
w[2]=w[2]+dw2
b1=b1+dv01
b2=b2+dv02
w[0]=w[0]+dw0
print("w=",w)
print("bias b1=",b1, " b2=",b2)
```

Output:

```
z1= 0.2
calculate net input to z2 layer
z2= 0.9
Apply activation function to calculate output
z1= 0.5498
z2= 0.7109
calculate net input to output layer
yin= 0.09101
calculate net output
y= 0.5227368084248941
dk 0.11906907074145694
compute error portion in delta
din1= 0.04762762829658278
din2= 0.011906907074145694
error in delta
fzin1 0.24751996
fzin2 0.20552119000000002
d1= 0.011788788650865037
d2= 0.0024471217110978417
Changes in weights between input and hidden layer
dv11= 0.0
dv21= 0.0029471971627162592
dv01= 0.0029471971627162592
dv12= 0.0
dv22= 0.0006117804277744604
dv02= 0.0006117804277744604
Final weights of network
v= [0.6 0.3]
v2 [-0.1  0.4]
w= [-0.17  0.42  0.12]
bias b1= 0.30294719716271623  b2= 0.5006117804277744
```

Practical 4b: Write a Program For Error Back Propagation Algorithm (Ebpa) Learning

```
import math
a0=-1
t=-1
w10=float(input("Enter weight first network"))
b10=float(input("Enter base first network:"))
w20=float(input("Enter weight second network:"))
b20=float(input("Enter base second network:"))
c=float(input("Enter learning coefficient:"))
n1=float(w10*c+b10)
a1=math.tanh(n1)
n2=float(w20*a1+b20)
a2=math.tanh(float(n2))
e=t-a2
s2=-2*(1-a2*a2)*e
s1=(1-a1*a1)*w20*s2

w21=w20-(c*s2*a1)
w11=w10-(c*s1*a0)
b21=b20-(c*s2)
b11=b10-(c*s1)
print("The updated weight of first n/w w11=",w11)
print("The uploaded weight of second n/w w21= ",w21)
print("The updated base of first n/w b10=",b10)
print("The updated base of second n/w b20= ",b20)
```

```
Enter weight first network:12
Enter base first network:35
Enter weight second network:23
Enter base second network:45
Enter learning coefficient:11
The updated weight of first n/w w11= 12.0
The uploaded weight of second n/w w21=  23.0
The updated base of first n/w b10= 35.0
The updated base of second n/w b20=  45.0
```

Practical 5

Practical 5a: Write a program for Hopfield Network.

```
#include "hop.h"
neuron::neuron(int *j)
{
inti;
for(i=0;i<4;i++)
    {
    weightv[i]= *(j+i);
    }
}
int neuron::act(int m, int *x)
{
inti;
int a=0;
for(i=0;i<m;i++)
    {
    a += x[i]*weightv[i];
    }
return a;
}
int network::threshld(int k)
{
if(k>=0)
    return (1);
else
    return (0);
}
network::network(int a[4],int b[4],int c[4],int d[4])
{
nrn[0] = neuron(a) ;
nrn[1] = neuron(b) ;
nrn[2] = neuron(c) ;
nrn[3] = neuron(d) ;
}
void network::activation(int *patrn)
{
inti,j;
for(i=0;i<4;i++)
    {
    for(j=0;j<4;j++)
        {
        cout<<"\n nrn["<<i<<"].weightv["<<j<<"] is "
            <<nrn[i].weightv[j];
        }
    nrn[i].activation = nrn[i].act(4,patrn);
    cout<<"\nactivation is "<<nrn[i].activation;
```

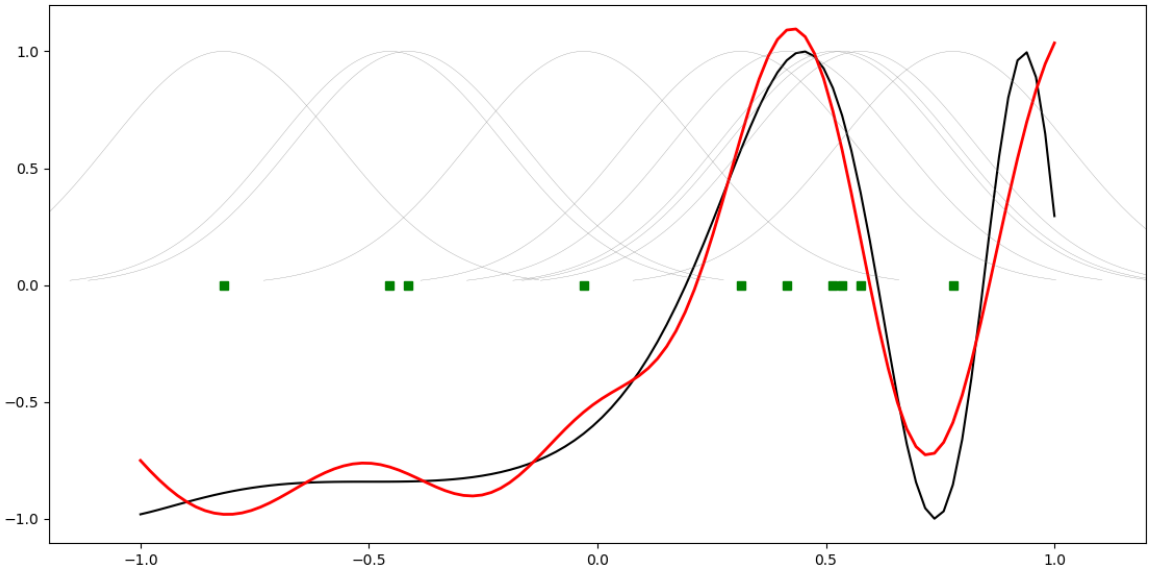
```
    output[i]=threshld(nrn[i].activation);
    cout<<"\noutput value is  "<<output[i]<<"\n";
    }
}
void main ()
{
int patrn1[]= {1,0,1,0},i;
int wt1[]= {0,-3,3,-3};
int wt2[]= {-3,0,-3,3};
int wt3[]= {3,-3,0,-3};
int wt4[]= {-3,3,-3,0};
cout<<"\nTHIS PROGRAM IS FOR A HOPFIELD NETWORK WITH A SINGLE LAYER
OF";
cout<<"\n4 FULLY INTERCONNECTED NEURONS. THE NETWORK SHOULD
RECALLTHE";
cout<<"\nPATTERNS 1010 AND 0101 CORRECTLY.\n";
//create the network by calling its constructor.
// the constructor calls neuron constructor as many times as thenumber of
// neurons in the network.
network h1(wt1,wt2,wt3,wt4);
//present a pattern to the network and get the activations of theneurons
h1.activation(patrn1);
//check if the pattern given is correctly recalled and give message
for(i=0;i<4;i++)
    {
    if (h1.output[i] == patrn1[i])
        cout<<"\n pattern= "<<patrn1[i]<<
        "  output = "<<h1.output[i]<<"  component matches";
    else
        cout<<"\n pattern= "<<patrn1[i]<<
        "  output = "<<h1.output[i]<<
        "  discrepancy occurred";
    }
cout<<"\n\n";
int patrn2[]= {0,1,0,1};
h1.activation(patrn2);
for(i=0;i<4;i++)
    {
    if (h1.output[i] == patrn2[i])
        cout<<"\n pattern= "<<patrn2[i]<<
        "  output = "<<h1.output[i]<<"  component matches";
    else
        cout<<"\n pattern= "<<patrn2[i]<<
        "  output = "<<h1.output[i]<<
        "  discrepancy occurred";
     }
}
======== End code of main program=============
```

```
//Hop.h
//Single layer Hopfield Network with 4 neurons
#include <stdio.h>
#include <iostream.h>
#include <math.h>
class neuron
{
protected:
    int activation;
    friend class network;
public:
    intweightv[4];
    neuron() {};
    neuron(int *j) ;
    int act(int, int*);
};
class network
{
public:
    neuron   nrn[4];
    int output[4];
    intthreshld(int) ;
    void activation(int j[4]);
    network(int*,int*,int*,int*);
};
```

Practical 5b: Write a program for Radial Basis function

```
from scipy import *
from scipy.linalg import norm, pinv
from matplotlib import pyplot as plt
class RBF:
    def __init__(self, indim, numCenters, outdim):
        self.indim =indim
        self.outdim =outdim
        self.numCenters =numCenters
        self.centers =[random.uniform(-1, 1, indim) for i in range(numCenters)]
        self.beta =8
        self.W =random.random((self.numCenters, self.outdim))
    def _basisfunc(self, c, d):
        assert len(d) ==self.indim
        return exp(-self.beta *norm(c-d)**2)
    def _calcAct(self, X):
        # calculate activations of RBFs
        G =zeros((X.shape[0], self.numCenters), float)
        for ci, c in enumerate(self.centers):
            for xi, x in enumerate(X):
                G[xi,ci] =self._basisfunc(c, x)
        return G
    def train(self, X, Y):
        """ X: matrix of dimensions n x indim
            y: column vector of dimension n x 1 """
        # choose random center vectors from training set
        rnd_idx =random.permutation(X.shape[0])[:self.numCenters]
        self.centers =[X[i,:] for i in rnd_idx]
        print("center", self.centers)
        # calculate activations of RBFs
        G =self._calcAct(X)
        print (G)
        # calculate output weights (pseudoinverse)
        self.W =dot(pinv(G), Y)
    def test(self, X):
        """ X: matrix of dimensions n x indim """
        G =self._calcAct(X)
        Y =dot(G, self.W)
        return Y
if __name__ =='__main__':
    # ----- 1D Example ------------------------------------------------
    n =100

    x =mgrid[-1:1:complex(0,n)].reshape(n, 1)
    # set y and add random noise
    y =sin(3*(x+0.5)**3-1)
    # y += random.normal(0, 0.1, y.shape)
    # rbf regression
```

```
rbf =RBF(1, 10, 1)
rbf.train(x, y)
z =rbf.test(x)
# plot original data
plt.figure(figsize=(12, 8))
plt.plot(x, y, 'k-')
# plot learned model
plt.plot(x, z, 'r-', linewidth=2)
# plot rbfs
plt.plot(rbf.centers, zeros(rbf.numCenters), 'gs')
for c in rbf.centers:
    # RF prediction lines
    cx =arange(c-0.7, c+0.7, 0.01)
    cy =[rbf._basisfunc(array([cx_]), array([c])) for cx_ in cx]
    plt.plot(cx, cy, '-', color='gray', linewidth=0.2)
plt.xlim(-1.2, 1.2)
plt.show()
```



Practical 6

Practical 6a: Self-Organizing Maps

The SOM algorithm is used to compress the information to produce a similarity graph while preserving the topologic relationship of the input data space.

The basic SOM model construction algorithm can be interpreted as follows:

1) Create and initialize a matrix (weight vector) randomly to hold the neurons. If the matrix can be initialized with order and roughly compiles with the input density function, the map will converge quickly

2) Read the input data space. For each observation (instance), use the optimum fit approach, which is based on the Euclidean distance

$c = arg_i \min \| x - m_i \|$

to find the neuron which best matches this observation. Let x denote the training vector from the observation and mi denote a single neuron in the matrix. Update that neuron to resemble that observation using the following equation:

$mi\ (t + 1) = mi\ (t) + h(t)[x(t) - mi\ (t)]$ (4) $mi(t)$:

the weight vector before the neuron is updated.

$(t + 1)$: the weight vector after the neuron is updated.

(t): the training vector from the observation.

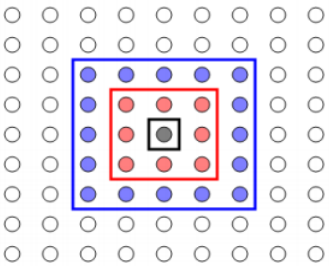
$h(t)$: the neighborhood function (a smoothing kernel defined over the lattice points), defined though the following equation:

$h(t) = \{\ \alpha(t), i \in Nc\ 0, i \in Nc$ (5)

: the neighborhood set, which decreases with time.

(t): the learning-rate factor which can be linear, exponential or inversely proportional.

It is a monotonically decreasing function of time (t)



In general, SOMs might be useful for visualizing high-dimensional data in terms of its similarity structure. Especially large SOMs (i.e. with large number of Kohonen units) are known to perform mappings that preserve the topology of the original data, i.e. neighboring data points in input space will also be represented in adjacent locations on the SOM.

The following code shows the 'classic' color mapping example, i.e. the SOM will map a number of colors into a rectangular area.

```
frommvpa2.suiteimport*
```

First, we define some colors as RGB values from the interval (0,1), i.e. with white being (1, 1, 1) and black being (0, 0, 0). Please note, that a substantial proportion of the defined colors represent variations of 'blue', which are supposed to be represented in more detail in the SOM.

```
colors=np.array(
[[0.,0.,0.],
[0.,0.,1.],
```

```
[0.,0.,0.5],
[0.125,0.529,1.0],
[0.33,0.4,0.67],
[0.6,0.5,1.0],
[0.,1.,0.],
[1.,0.,0.],
[0.,1.,1.],
[1.,0.,1.],
[1.,1.,0.],
[1.,1.,1.],
[.33,.33,.33],
[.5,.5,.5],
[.66,.66,.66]])

# store the names of the colors for visualization later on
color_names= \
['black','blue','darkblue','skyblue',
'greyblue','lilac','green','red',
'cyan','violet','yellow','white',
'darkgrey','mediumgrey','lightgrey']
```

Now we can instantiate the mapper. It will internally use a so-called Kohonen layer to map the data onto. We tell the mapper to use a rectangular layer with 20 x 30 units. This will be the output space of the mapper. Additionally, we tell it to train the network using 400 iterations and to use custom learning rate.

```
som=SimpleSOMMapper((20,30),400,learning_rate=0.05)
```

Finally, we train the mapper with the previously defined ‘color’ dataset.

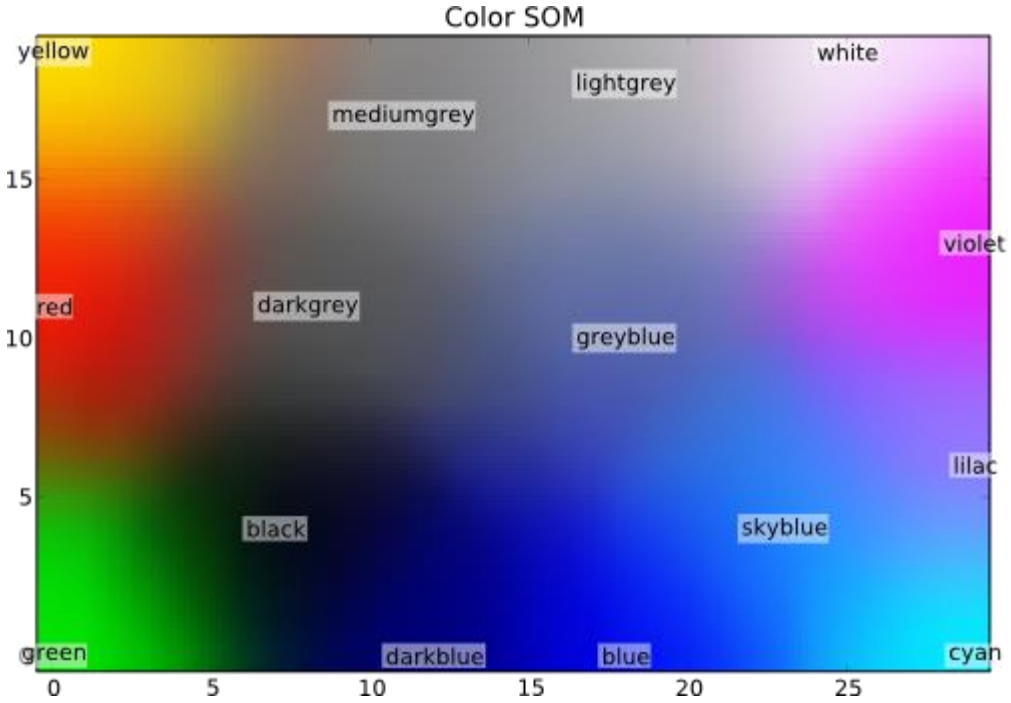
```
som.train(colors)
```

Each unit in the Kohonen layer can be treated as a pointer into the high-dimensional input space, that can be queried to inspect which input subspaces the SOM maps onto certain sections of its 2D output space. The color-mapping generated by this example’s SOM can be shown with a single matplotlib call:

```
pl.imshow(som.K,origin='lower')
```

And now, let’s take a look onto which coordinates the initial training prototypes were mapped to. The get those coordinates we can simply feed the training data to the mapper and plot the output.

```
mapped=som(colors)

pl.title('Color SOM')
# SOM's kshape is (rows x columns), while matplotlib wants (X x Y)
fori,m in enumerate(mapped):
pl.text(m[1],m[0],color_names[i],ha='center',va='center',
bbox=dict(facecolor='white',alpha=0.5,lw=0))
```


Color SOM
yellow
white
lightgrey
mediumgrey
15
violet
red
darkgrey
10
greyblue
lilac
5
black
skyblue
green
darkblue
blue
cyan
0
5
10
15
20
25

PRACTICAL 6B: ADAPTIVE RESONANCE THEORY

```
from__future__import
division

importnumpyas np
fromneupy.utilsimportformat_data
fromneupy.core.propertiesimport (ProperFractionProperty,
IntProperty)
fromneupy.algorithms.baseimportBaseNetwork
__all__= ('ART1',)
classART1(BaseNetwork):
"""
    Adaptive Resonance Theory (ART1) Network for binary
    data clustering.
    Notes
    -----
    - Weights are not random, so the result will be
      always reproduceble.
    Parameters
    ----------
    rho : float
        Control reset action in training process. Value must be
        between ``0`` and ``1``, defaults to ``0.5``.
n_clusters : int
        Number of clusters, defaults to ``2``. Min value is also
``2``.
    {BaseNetwork.Parameters}
    Methods
    -------
    train(X)
        ART trains until all clusters are found.
    predict(X)
        Each prediction trains a new network. It's an alias to
        the ``train`` method.
    {BaseSkeleton.fit}
    Examples
    --------
>>>import numpy as np
>>>from neupy import algorithms
>>>
>>>data = np.array([
...     [0, 1, 0],
...     [1, 0, 0],
...     [1, 1, 0],
... ])
>>>>
```

```
>>>artnet = algorithms.ART1(
...     step=2,
...     rho=0.7,
... n_clusters=2,
...     verbose=False
... )
>>>artnet.predict(data)
   array([ 0.,  1.,  1.])
   """
   rho =ProperFractionProperty(default=0.5)
n_clusters=IntProperty(default=2, minval=2)
deftrain(self, X):
       X =format_data(X)
ifX.ndim!=2:
raiseValueError("Input value must be 2 dimensional, got "
"{}".format(X.ndim))
nsamples, n_features=X.shape
n_clusters=self.n_clusters
       step =self.step
       rho =self.rho
ifnp.any((X !=0) & (X !=1)):
raiseValueError("ART1 Network works only with binary
matrices")
ifnothasattr(self, 'weight_21'):
self.weight_21 =np.ones((n_features, n_clusters))
ifnothasattr(self, 'weight_12'):
          scaler = step / (step +n_clusters-1)
self.weight_12 = scaler *self.weight_21.T
       weight_21 =self.weight_21
       weight_12 =self.weight_12
ifn_features!= weight_21.shape[0]:
raiseValueError("Input data has invalid number of features. "
"Got {} instead of {}"
"".format(n_features, weight_21.shape[0]))
       classes =np.zeros(n_samples)
# Train network
fori, p inenumerate(X):
disabled_neurons= []
reseted_values= []
          reset =True
while reset:
             output1 = p
             input2 = np.dot(weight_12, output1.T)
             output2 =np.zeros(input2.size)
             input2[disabled_neurons] =-np.inf
winner_index= input2.argmax()
             output2[winner_index] =1
             expectation = np.dot(weight_21, output2)
             output1 =np.logical_and(p, expectation).astype(int)
```

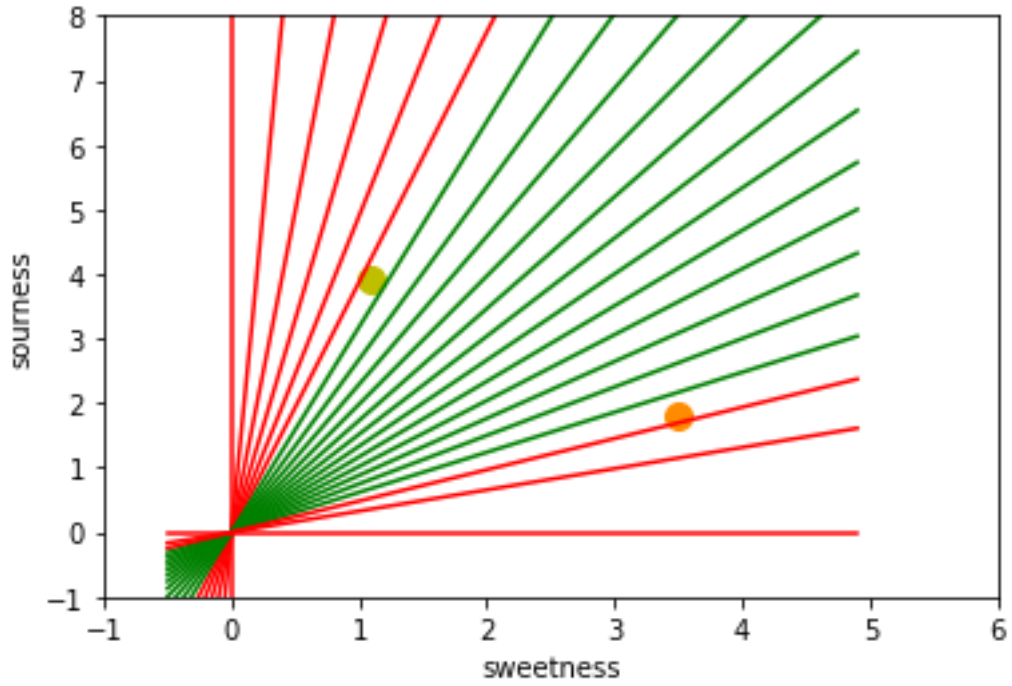
```
reset_value= np.dot(output1.T, output1) / np.dot(p.T, p)
            reset =reset_value< rho
if reset:
disabled_neurons.append(winner_index)
reseted_values.append((reset_value, winner_index))
iflen(disabled_neurons) >=n_clusters:
# Got this case only if we test all possible clusters
                reset =False
winner_index=None
ifnot reset:
ifwinner_indexisnotNone:
                    weight_12[winner_index, :] = (step * output1) / (
                       step + np.dot(output1.T, output1) -1
                    )
                    weight_21[:, winner_index] = output1
else:
# Get result with the best `rho`
winner_index=max(reseted_values)[1]
                classes[i] =winner_index
return classes
defpredict(self, X):
returnself.train(X)
```

Practical 7

Practical 7a: Line Separation

```
import numpy as np
import matplotlib.pyplot as plt
def create_distance_function(a, b, c):
    """ 0 = ax + by + c """
    def distance(x, y):
        """ returns tuple (d, pos)
            d is the distance
            If pos == -1 point is below the line,
            0 on the line and +1 if above the line
        """
        nom = a * x + b * y + c
        if nom == 0:
            pos = 0
        elif (nom<0 and b<0) or (nom>0 and b>0):
            pos = -1
        else:
            pos = 1
        return (np.absolute(nom) / np.sqrt( a ** 2 + b ** 2), pos)
    return distance

points = [ (3.5, 1.8), (1.1, 3.9) ]
fig, ax = plt.subplots()
ax.set_xlabel("sweetness")
ax.set_ylabel("sourness")
ax.set_xlim([-1, 6])
ax.set_ylim([-1, 8])
X = np.arange(-0.5, 5, 0.1)
colors = ["r", ""] # for the samples
size = 10
for (index, (x, y)) in enumerate(points):
    if index== 0:
        ax.plot(x, y, "o", color="darkorange", markersize=size)
    else:
        ax.plot(x, y, "oy", markersize=size)
        step = 0.05
for x in np.arange(0, 1+step, step):
    slope = np.tan(np.arccos(x))
    dist4line1 = create_distance_function(slope, -1, 0)
    #print("x: ", x, "slope: ", slope)
    Y = slope * X

    results = []
    for point in points:
        results.append(dist4line1(*point))
    #print(slope, results)
    if (results[0][1] != results[1][1]):
```

```
        ax.plot(X, Y, "g-")
    else:
        ax.plot(X, Y, "r-")

plt.show()
```



Practical 8

Practical 8a: Membership and Identity operators in, not in.

```
# Python program to illustrate
# Finding common member in list
# without using 'in' operator

# Define a function() that takes two lists
def overlapping(list1,list2):
	c=0
	d=0
	for i in list1:
		c+=1
	for i in list2:
		d+=1
	for i in range(0,c):
		for j in range(0,d):
			if(list1[i]==list2[j]):
				return 1
	return 0
list1=[1,2,3,4,5]
list2=[6,7,8,9]
if(overlapping(list1,list2)):
	print("overlapping")
else:
	print("not overlapping")
```

```
# Python program to illustrate
# Finding common member in list
# without using 'in' operator

# Define a function() that takes two lists
def overlapping(list1,list2):

	c=0
	d=0
	for i in list1:
		c+=1
	for i in list2:
		d+=1
	for i in range(0,c):
		for j in range(0,d):
			if(list1[i]==list2[j]):
				return 1
	return 0
list1=[1,2,3,4,5]
list2=[6,7,8,9]
```

```
if(overlapping(list1,list2)):
        print("overlapping")
else:
        print("not overlapping")
```

Practical 8b: Membership and Identity Operators is, is not

```
# Python program to illustrate the use
# of 'is' identity operator
x = 5
if (type(x) is int):
        print ("true")
else:
        print ("false")
```

```
# Python program to illustrate the
# use of 'is not' identity operator
x = 5.2
if (type(x) is not int):
        print ("true")
else:
        print ("false")
```

Practical 9

Practical 9a: Find the ratios using fuzzy logic

pip install fuzzywuzzy
pip install python-Levenshtein

```
# Python code showing all the ratios together,
# make sure you have installed fuzzywuzzy module

from fuzzywuzzy import fuzz
from fuzzywuzzy import process
s1 = "I love fuzzysforfuzzys"
s2 = "I am loving fuzzysforfuzzys"
print ("FuzzyWuzzy Ratio:", fuzz.ratio(s1, s2))
print ("FuzzyWuzzyPartialRatio: ", fuzz.partial_ratio(s1, s2))
print ("FuzzyWuzzyTokenSortRatio: ", fuzz.token_sort_ratio(s1, s2))
print ("FuzzyWuzzyTokenSetRatio: ", fuzz.token_set_ratio(s1, s2))
print ("FuzzyWuzzyWRatio: ", fuzz.WRatio(s1, s2),'\n\n')
# for process library,
query = 'fuzzys for fuzzys'
choices = ['fuzzy for fuzzy', 'fuzzy fuzzy', 'g. for fuzzys']
print ("List of ratios: ")
print (process.extract(query, choices), '\n')
print ("Best among the above list: ",process.extractOne(query, choices))
```

Practical 9b: Solve Tipping Problem using fuzzy logic

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

# New Antecedent/Consequent objects hold universe variables and membership
# functions
quality = ctrl.Antecedent(np.arange(0, 11, 1), 'quality')
service = ctrl.Antecedent(np.arange(0, 11, 1), 'service')
tip = ctrl.Consequent(np.arange(0, 26, 1), 'tip')

# Auto-membership function population is possible with .automf(3, 5, or 7)
quality.automf(3)
service.automf(3)

# Custom membership functions can be built interactively with a familiar,
# Pythonic API
tip['low'] = fuzz.trimf(tip.universe, [0, 0, 13])
tip['medium'] = fuzz.trimf(tip.universe, [0, 13, 25])
tip['high'] = fuzz.trimf(tip.universe, [13, 25, 25])

"""
To help understand what the membership looks like, use the ``view`` methods.
"""

# You can see how these look with .view()
quality['average'].view()
"""
.. image:: PLOT2RST.current_figure
"""
service.view()
"""
.. image:: PLOT2RST.current_figure
"""
tip.view()
"""
.. image:: PLOT2RST.current_figure
```

Practical 10

Practical 10: Implementation of simple genetic algorithm

```
import random
# Number of individuals in each generation
POPULATION_SIZE =100
# Valid genes
GENES ='"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOP
QRSTUVWXYZ 1234567890, .-;:_!"#%&/()=?@${[]}'"
# Target string to be generated
TARGET ="I love GeeksforGeeks"
class Individual(object):
    '''
    Class representing individual in population
    '''
    def __init__(self, chromosome):
        self.chromosome =chromosome
        self.fitness =self.cal_fitness()
    @classmethod
    def mutated_genes(self):
        '''
        create random genes for mutation
        '''
        global GENES
        gene =random.choice(GENES)
        return gene
    @classmethod
    def create_gnome(self):
        '''
        create chromosome or string of genes
        '''
        global TARGET
        gnome_len =len(TARGET)
        return[self.mutated_genes() for _ in range(gnome_len)]
    def mate(self, par2):
        '''
        Perform mating and produce new offspring
        '''
        # chromosome for offspring
        child_chromosome =[]
        for gp1, gp2 in zip(self.chromosome, par2.chromosome):

            # random probability
            prob =random.random()

            # if prob is less than 0.45, insert gene
            # from parent 1
            if prob< 0.45:
```

```
            child_chromosome.append(gp1)

        # if prob is between 0.45 and 0.90, insert
        # gene from parent 2
        elif prob< 0.90:
            child_chromosome.append(gp2)

        # otherwise insert random gene(mutate),
        # for maintaining diversity
        else:
            child_chromosome.append(self.mutated_genes())

    # create new Individual(offspring) using
    # generated chromosome for offspring
    return Individual(child_chromosome)

def cal_fitness(self):
    '''
    Calculate fittness score, it is the number of
    characters in string which differ from target
    string.
    '''
    global TARGET
    fitness =0
    for gs, gt in zip(self.chromosome, TARGET):
        if gs !=gt: fitness+=1
    return fitness

# Driver code
def main():
    global POPULATION_SIZE

    #current generation
    generation =1

    found =False
    population =[]

    # create initial population
    for _ in range(POPULATION_SIZE):
            gnome =Individual.create_gnome()
            population.append(Individual(gnome))

    while not found:

        # sort the population in increasing order of fitness score
        population =sorted(population, key =lambda x:x.fitness)

        # if the individual having lowest fitness score ie.
```

```
        # 0 then we know that we have reached to the target
        # and break the loop
        if population[0].fitness <=0:
            found =True
            break

        # Otherwise generate new offsprings for new generation
        new_generation =[]

        # Perform Elitism, that mean 10% of fittest population
        # goes to the next generation
        s =int((10*POPULATION_SIZE)/100)
        new_generation.extend(population[:s])

        # From 50% of fittest population, Individuals
        # will mate to produce offspring
        s =int((90*POPULATION_SIZE)/100)
        for _ in range(s):
            parent1 =random.choice(population[:50])
            parent2 =random.choice(population[:50])
            child =parent1.mate(parent2)
            new_generation.append(child)

        population =new_generation

        print("Generation: {}\tString: {}\tFitness:
{}".format(generation,"".join(population[0].chromosome),population[0].fitness))
        generation +=1


    print("Generation: {}\tString: {}\tFitness: {}".format(generation,
"".join(population[0].chromosome),
          population[0].fitness))

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

Output:

```
Generation: 1    String: tO{"-?=jH[k8=B4]Oe@}    Fitness: 18
Generation: 2    String: tO{"-?=jH[k8=B4]Oe@}    Fitness: 18
Generation: 3    String: .#lRWf9k_Ifslw #O$k_    Fitness: 17
Generation: 4    String: .-1Rq?9mHqk3Wo]3rek_    Fitness: 16
Generation: 5    String: .-1Rq?9mHqk3Wo]3rek_    Fitness: 16
Generation: 6    String: A#ldW) #lIkslwcVek)    Fitness: 14
Generation: 7    String: A#ldW) #lIkslwcVek)    Fitness: 14
Generation: 8    String: (, o x _x%Rs=, 6Peek3   Fitness: 13
                    .
                    .
```

.

```
Generation: 29    String: I lope Geeks#o, Geeks    Fitness: 3
Generation: 30    String: I loMeGeeksfoBGeeks    Fitness: 2
Generation: 31    String: I love Geeksfo0Geeks    Fitness: 1
Generation: 32    String: I love Geeksfo0Geeks    Fitness: 1
Generation: 33    String: I love Geeksfo0Geeks    Fitness: 1
Generation: 34    String: I love GeeksforGeeks    Fitness: 0
```

Practical 10 b: Create two classes: City and Fitness using Genetic algorithm

first create a City class that will allow us to create and handle our cities.

Create Population

```
import numpy as np, random, operator, pandas as pd, matplotlib.pyplot as plt
from tkinter import Tk, Canvas, Frame, BOTH, Text
import math
class City:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def distance(self, city):
        xDis = abs(self.x - city.x)
        yDis = abs(self.y - city.y)
        distance = np.sqrt((xDis ** 2) + (yDis ** 2))
        return distance
    def __repr__(self):
        return "(" + str(self.x) + "," + str(self.y) + ")"
class Fitness:
    def __init__(self, route):
        self.route = route
        self.distance = 0
        self.fitness= 0.0
    def routeDistance(self):
        if self.distance ==0:
            pathDistance = 0
            for i in range(0, len(self.route)):
                fromCity = self.route[i]
                toCity = None
                if i + 1 < len(self.route):
                    toCity = self.route[i + 1]
                else:
                    toCity = self.route[0]
                pathDistance += fromCity.distance(toCity)
            self.distance = pathDistance
        return self.distance
    def routeFitness(self):
        if self.fitness == 0:
            self.fitness = 1 / float(self.routeDistance())
        return self.fitness
def createRoute(cityList):
```

```
    route = random.sample(cityList, len(cityList))
    return route
def initialPopulation(popSize, cityList):
    population = []
    for i in range(0, popSize):
        population.append(createRoute(cityList))
    return population
def rankRoutes(population):
    fitnessResults = {}
    for i in range(0,len(population)):
        fitnessResults[i] = Fitness(population[i]).routeFitness()
    return sorted(fitnessResults.items(), key = operator.itemgetter(1), reverse = True)
def selection(popRanked, eliteSize):
    selectionResults = []
    df = pd.DataFrame(np.array(popRanked), columns=["Index","Fitness"])
    df['cum_sum'] = df.Fitness.cumsum()
    df['cum_perc'] = 100*df.cum_sum/df.Fitness.sum()
    for i in range(0, eliteSize):
        selectionResults.append(popRanked[i][0])
    for i in range(0, len(popRanked) - eliteSize):
        pick = 100*random.random()
        for i in range(0, len(popRanked)):
            if pick <= df.iat[i,3]:
                selectionResults.append(popRanked[i][0])
                break
    return selectionResults
def matingPool(population, selectionResults):
    matingpool = []
    for i in range(0, len(selectionResults)):
        index = selectionResults[i]
        matingpool.append(population[index])
    return matingpool
def breed(parent1, parent2):
    child = []
    childP1 = []
    childP2 = []
    geneA = int(random.random() * len(parent1))
    geneB = int(random.random() * len(parent1))
    startGene = min(geneA, geneB)
    endGene = max(geneA, geneB)
    for i in range(startGene, endGene):
        childP1.append(parent1[i])
    childP2 = [item for item in parent2 if item not in childP1]
    child = childP1 + childP2
    return child
def breedPopulation(matingpool, eliteSize):
    children = []
    length = len(matingpool) - eliteSize
    pool = random.sample(matingpool, len(matingpool))
```

```
    for i in range(0,eliteSize):
        children.append(matingpool[i])
    for i in range(0, length):
        child = breed(pool[i], pool[len(matingpool)-i-1])
        children.append(child)
    return children
def mutate(individual, mutationRate):
    for swapped in range(len(individual)):
        if(random.random() < mutationRate):
            swapWith = int(random.random() * len(individual))
            city1 = individual[swapped]
            city2 = individual[swapWith]
            individual[swapped] = city2
            individual[swapWith] = city1
    return individual
def mutatePopulation(population, mutationRate):
    mutatedPop = []
    for ind in range(0, len(population)):
        mutatedInd = mutate(population[ind], mutationRate)
        mutatedPop.append(mutatedInd)
    return mutatedPop
def nextGeneration(currentGen, eliteSize, mutationRate):
    popRanked = rankRoutes(currentGen)
    selectionResults = selection(popRanked, eliteSize)
    matingpool = matingPool(currentGen, selectionResults)
    children = breedPopulation(matingpool, eliteSize)
    nextGeneration = mutatePopulation(children, mutationRate)
    return nextGeneration
def geneticAlgorithm(population, popSize, eliteSize, mutationRate, generations):
    pop = initialPopulation(popSize, population)
    print("Initial distance: " + str(1 / rankRoutes(pop)[0][1]))
    for i in range(0, generations):
        pop = nextGeneration(pop, eliteSize, mutationRate)
    print("Final distance: " + str(1 / rankRoutes(pop)[0][1]))
    bestRouteIndex = rankRoutes(pop)[0][0]
    bestRoute = pop[bestRouteIndex]
    return bestRoute
def geneticAlgorithmPlot(population, popSize, eliteSize, mutationRate, generations):
    pop = initialPopulation(popSize, population)
    progress = []
    progress.append(1 / rankRoutes(pop)[0][1])
    for i in range(0, generations):
        pop = nextGeneration(pop, eliteSize, mutationRate)
        progress.append(1 / rankRoutes(pop)[0][1])
    plt.plot(progress)
    plt.ylabel('Distance')
    plt.xlabel('Generation')
    plt.show()
def main():
```

```
    cityList = []
    for i in range(0,25):
        cityList.append(City(x=int(random.random() * 200), y=int(random.random() * 200)))
    geneticAlgorithmPlot(population=cityList, popSize=100, eliteSize=20,
mutationRate=0.01, generations=500)
if __name__ == '__main__':
    main()
```