Index

EXPERIMENT NO: 1

Aim: To write a Java program that demonstrates the implementation of Remote Procedure Call (RPC).

Code:

The client successfully invokes a remote procedure on the server, demonstrating the RPC mechanism.

1️⃣ Addition.java

```
import java.rmi.Remote;

import java.rmi.RemoteException;


public interface Addition extends Remote {

    int addNumbers(int a, int b) throws RemoteException;

}
```

2️⃣ AdditionImpl.java

```
import java.rmi.server.UnicastRemoteObject;

import java.rmi.RemoteException;


public class AdditionImpl extends UnicastRemoteObject implements Addition {

    protected AdditionImpl() throws RemoteException {

        super();

    }


    public int addNumbers(int a, int b) throws RemoteException {

        System.out.println("Processing request for addition: " + a + " + " + b);

        return a + b;

    }
}
```

3️⃣ Server.java

```
import java.rmi.Naming;

public class Server {
    public static void main(String[] args) {
        try {
            AdditionImpl obj = new AdditionImpl();
            Naming.rebind("rmi://localhost:5000/additionService", obj);
            System.out.println("Server is ready and waiting for client requests...");
        } catch (Exception e) {
            System.out.println("Server error: " + e);
        }
    }
}
```

4️⃣ Client.java

```
import java.rmi.Naming;
import java.util.Scanner;

public class Client {
    public static void main(String[] args) {
        try {
            Addition stub = (Addition) Naming.lookup("rmi://localhost:5000/additionService");
            Scanner sc = new Scanner(System.in);

            System.out.print("Enter first number: ");
            int a = sc.nextInt();
            System.out.print("Enter second number: ");
            int b = sc.nextInt();

            int result = stub.addNumbers(a, b);
            System.out.println("Result received from Server: " + result);
        } catch (Exception e) {
            System.out.println("Client error: " + e);
        }
    }
}
```

Output:

Server terminal

```
C:\rmi_project>javac *.java
C:\rmi_project>start rmiregistry
C:\rmi_project>java Server
Server is ready and waiting for client requests...
Processing request for addition: 15 + 25
```

Client terminal

```
C:\rmi_project>java Client
Enter first number: 15
Enter second number: 25
Result received from Server: 40
```

EXPERIMENT NO: 2

Aim: To implement the concept of Remote Method Invocation (RMI) in Java.

Code:

1️⃣ Remote Interface – Hello.java

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


// Remote Interface
public interface Hello extends Remote {
    String sayHello(String name) throws RemoteException;
}
```

2️⃣ Implementation – HelloImpl.java

```
import java.rmi.server.UnicastRemoteObject;
import java.rmi.RemoteException;


// Implementation of the Remote Interface
public class HelloImpl extends UnicastRemoteObject implements Hello {


    protected HelloImpl() throws RemoteException {
        super();
    }


    public String sayHello(String name) throws RemoteException {
        System.out.println("Received request from client for: " + name);
        return "Hello, " + name + "! Welcome to Java RMI.";
    }
}
```

3 Server – RMIServer.java

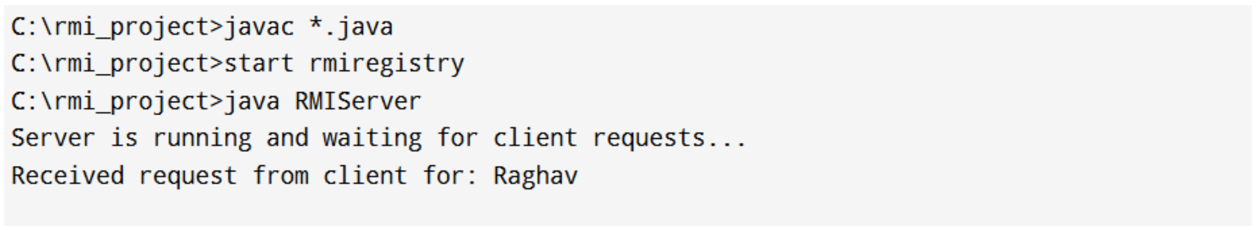
```
import java.rmi.Naming;

public class RMIServer {
    public static void main(String[] args) {
        try {
            HelloImpl obj = new HelloImpl();
            Naming.rebind("rmi://localhost:5099/helloService", obj);
            System.out.println("Server is running and waiting for client requests...");
        } catch (Exception e) {
            System.out.println("Server Exception: " + e);
        }
    }
}
```

4 Client – RMIClient.java

```
import java.rmi.Naming;
import java.util.Scanner;

public class RMIClient {
    public static void main(String[] args) {
        try {
            Hello stub = (Hello) Naming.lookup("rmi://localhost:5099/helloService");
            Scanner sc = new Scanner(System.in);

            System.out.print("Enter your name: ");
            String name = sc.nextLine();


            String response = stub.sayHello(name);
            System.out.println("Response from Server: " + response);
        } catch (Exception e) {
            System.out.println("Client Exception: " + e);
        }
    }
}
```

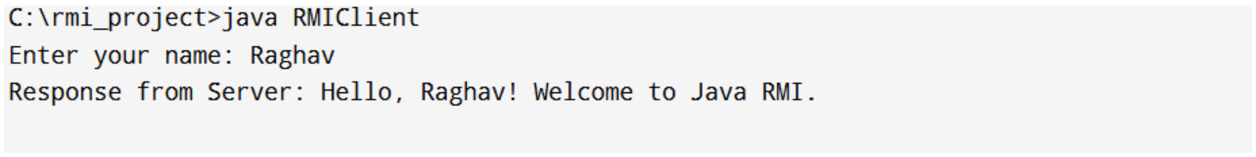
Output:

Server terminal



```
C:\rmi_project>javac *.java
C:\rmi_project>start rmiregistry
C:\rmi_project>java RMIServer
Server is running and waiting for client requests...
Received request from client for: Raghav
```

Client terminal



```
C:\rmi_project>java RMIClient
Enter your name: Raghav
Response from Server: Hello, Raghav! Welcome to Java RMI.
```

EXPERIMENT NO: 3

Aim: To implement Lamport's Logical Clock in Java for event ordering in distributed systems.

Code:

LamportClock.java

```
import java.util.Scanner;


public class LamportClock {


    // Function to find maximum of two integers
    public static int max(int a, int b) {
        return (a > b) ? a : b;
    }


    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);


        // Input number of events in Process 1 and Process 2
        System.out.print("Enter number of events in Process 1: ");
        int n1 = sc.nextInt();
        System.out.print("Enter number of events in Process 2: ");
        int n2 = sc.nextInt();


        int[] p1 = new int[n1];
        int[] p2 = new int[n2];
        // Initialize clocks
        for (int i = 0; i < n1; i++) {
            p1[i] = i + 1;
        }
        for (int i = 0; i < n2; i++) {
            p2[i] = i + 1;
        }

        System.out.print("Enter number of messages sent between processes: ");
        int m = sc.nextInt();
        int[][] messages = new int[m][2];

        // Input message pairs (sendEvent, receiveEvent)
        System.out.println("Enter message pairs (sendEventNo in P1 or P2, receiveEventNo in
other process):");
        for (int i = 0; i < m; i++) {
            System.out.print("Message " + (i + 1) + ": ");
            messages[i][0] = sc.nextInt(); // sender event index
            messages[i][1] = sc.nextInt(); // receiver event index
        }

        // Apply Lamport clock rules
        for (int i = 0; i < m; i++) {
            int sendEvent = messages[i][0];
            int recvEvent = messages[i][1];

            // Assuming messages from P1 to P2 for simplicity
            p2[recvEvent - 1] = max(p2[recvEvent - 1], p1[sendEvent - 1]) + 1;

            // Update all later events in P2
            for (int j = recvEvent; j < n2; j++) {
                p2[j] = p2[j - 1] + 1;
            }
        }

        // Display Lamport timestamps
        System.out.println("\n--- Lamport Logical Clocks ---");
        System.out.print("Process 1: ");
        for (int i = 0; i < n1; i++) {
            System.out.print(p1[i] + " ");
        }

        System.out.print("\nProcess 2: ");
        for (int i = 0; i < n2; i++) {
            System.out.print(p2[i] + " ");
        }

        System.out.println("\n");
        sc.close();
    }
}
```

Output:

```
C:\lamport>javac LamportClock.java
C:\lamport>java LamportClock
Enter number of events in Process 1: 3
Enter number of events in Process 2: 3
Enter number of messages sent between processes: 1
Enter message pairs (sendEventNo in P1 or P2, receiveEventNo in other process):
Message 1: 2 1

--- Lamport Logical Clocks ---
Process 1: 1 2 3
Process 2: 3 4 5
```

EXPERIMENT NO: 4

Aim: To implement a mutual exclusion service using Lamport's Mutual Exclusion Algorithm in a distributed system.

Code:

LamportMutex.java

```
import java.util.*;


class LamportProcess {
    int pid;
    int clock = 0;
    boolean requestingCS = false;
    PriorityQueue<Request> requestQueue = new PriorityQueue<>(Comparator.comparingInt(r -
> r.timestamp));


    public LamportProcess(int pid) {
        this.pid = pid;
    }


    public void requestCS(List<LamportProcess> allProcesses) {
        requestingCS = true;
        clock++;
        int timestamp = clock;


        System.out.println("\nProcess " + pid + " requesting CS with timestamp " + timestamp);
        for (LamportProcess p : allProcesses) {
            if (p.pid != this.pid)
                p.receiveRequest(timestamp, pid);
        }

        // Wait for replies from all others (simulated)
        System.out.println("Process " + pid + " received all replies. Entering CS...");
        enterCS(allProcesses);
    }

    public void receiveRequest(int timestamp, int senderPid) {
        clock = Math.max(clock, timestamp) + 1;
        requestQueue.add(new Request(timestamp, senderPid));
        System.out.println("Process " + pid + " received REQUEST from P" + senderPid + " (TS="
+ timestamp + ")");
        // Immediately send REPLY (simulated)
    }

    public void enterCS(List<LamportProcess> allProcesses) {
        System.out.println(">>> Process " + pid + " ENTERED critical section.");
        clock++;

        // Simulate CS execution
        try { Thread.sleep(500); } catch (InterruptedException e) { }

        exitCS(allProcesses);
    }
    public void exitCS(List<LamportProcess> allProcesses) {
        requestingCS = false;
        clock++;
        System.out.println("<<< Process " + pid + " EXITING critical section.");

        for (LamportProcess p : allProcesses) {
            if (p.pid != this.pid)
                p.receiveRelease(pid);
        }
    }

    public void receiveRelease(int senderPid) {
        requestQueue.removeIf(r -> r.pid == senderPid);
        System.out.println("Process " + pid + " received RELEASE from P" + senderPid);
    }
}

class Request {
    int timestamp;
    int pid;

    public Request(int timestamp, int pid) {
        this.timestamp = timestamp;
        this.pid = pid;
    }
}
```

```
public class LamportMutex {
    public static void main(String[] args) {
        List<LamportProcess> processes = new ArrayList<>();
        for (int i = 1; i <= 3; i++) {
            processes.add(new LamportProcess(i));
        }

        // Simulate process 1 requesting CS first, then process 2
        processes.get(0).requestCS(processes);
        processes.get(1).requestCS(processes);
        processes.get(2).requestCS(processes);
    }
}
```

Output:

```
C:\lamport_mutex>javac LamportMutex.java
C:\lamport_mutex>java LamportMutex

Process 1 requesting CS with timestamp 1
Process 2 received REQUEST from P1 (TS=1)
Process 3 received REQUEST from P1 (TS=1)
Process 1 received all replies. Entering CS...
>>> Process 1 ENTERED critical section.
<<< Process 1 EXITING critical section.
Process 2 received RELEASE from P1
Process 3 received RELEASE from P1
```

```
Process 2 requesting CS with timestamp 2
Process 1 received REQUEST from P2 (TS=2)
Process 3 received REQUEST from P2 (TS=2)
Process 2 received all replies. Entering CS...
>>> Process 2 ENTERED critical section.
<<< Process 2 EXITING critical section.
Process 1 received RELEASE from P2
Process 3 received RELEASE from P2

Process 3 requesting CS with timestamp 3
Process 1 received REQUEST from P3 (TS=3)
Process 2 received REQUEST from P3 (TS=3)
Process 3 received all replies. Entering CS...
>>> Process 3 ENTERED critical section.
<<< Process 3 EXITING critical section.
Process 1 received RELEASE from P3
Process 2 received RELEASE from P3
```

EXPERIMENT NO: 5

Aim: To install and configure Hadoop on a Windows operating system for big data processing.

Theory:

Hadoop is an open-source framework that enables distributed storage and processing of large datasets across clusters of computers using simple programming models. It consists of two main components:
1. Hadoop Distributed File System (HDFS): A distributed file system that provides high throughput access to application data.
2. YARN (Yet Another Resource Negotiator): A resource management platform responsible for managing compute resources in clusters and scheduling users' applications.

Hadoop can be installed on a variety of platforms, including Windows, although it is traditionally used on Linux systems. The installation on Windows involves setting up Java, configuring environment variables, and ensuring that the necessary components are correctly installed and configured.

Procedure:

1. Install Java SDK:
   - Download the latest version of JDK from Oracle's website.
   - Install JDK and set the JAVA_HOME environment variable in the system properties.

2. Download Hadoop:
   - Visit the Apache Hadoop website and download the appropriate version of Hadoop.
   - Extract the downloaded archive to a suitable directory on your system.

3. Set Environment Variables:
   - Set HADOOP_HOME to the Hadoop installation directory.
   - Add %HADOOP_HOME%\bin to the system PATH variable.

4. Configure Hadoop:
   - Navigate to the etc/hadoop directory and configure the following files:
     - core-site.xml: Set the default file system and path.
     - hdfs-site.xml: Configure the replication factor and data node directories.

   - mapred-site.xml: Define the job tracker and task tracker.
   - yarn-site.xml: Set resource manager and node manager settings.

5. Format HDFS:
  - Use the command hdfs namenode -format to format the Hadoop filesystem.

6. Start Hadoop Services:
  - Start the NameNode and DataNode using start-dfs.cmd.
  - Start the ResourceManager and NodeManager using start-yarn.cmd.

7. Verification:
  - Access the Hadoop web interfaces for HDFS and YARN to verify the installation and configuration.

Output:

```
C:\hadoop>hdfs namenode -format
2025-10-29 18:45:10 INFO namenode.NameNode: Formatting successful.

C:\hadoop>start-dfs.cmd
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes
localhost: starting datanode, logging to C:\hadoop\logs\hadoop-datanode.log

C:\hadoop>start-yarn.cmd
starting resourcemanager, logging to C:\hadoop\logs\hadoop-resourcemanager.log
starting nodemanager, logging to C:\hadoop\logs\hadoop-nodemanager.log
```

EXPERIMENT NO: 6

Aim: To run a simple MapReduce application on a single-node Hadoop cluster for data processing.

Code:

WordCount.java

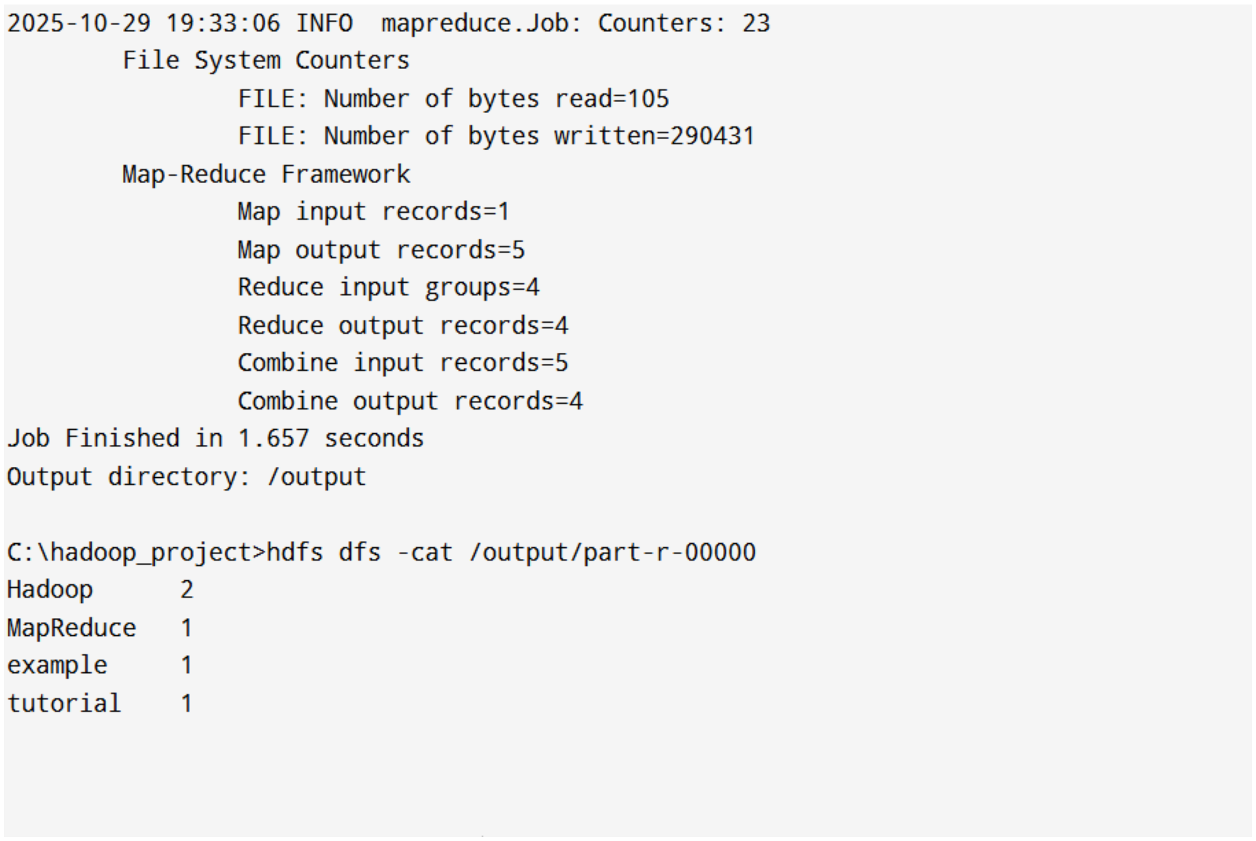
```
import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


public class WordCount {

  public static class TokenizerMapper

     extends Mapper<Object, Text, Text, IntWritable> {
```

```
    private final static IntWritable one = new IntWritable(1);

    private Text word = new Text();



    public void map(Object key, Text value, Context context)

            throws IOException, InterruptedException {

        StringTokenizer itr = new StringTokenizer(value.toString());

        while (itr.hasMoreTokens()) {

            word.set(itr.nextToken());

            context.write(word, one);

        }

    }

}



public static class IntSumReducer

     extends Reducer<Text, IntWritable, Text, IntWritable> {



    private IntWritable result = new IntWritable();



    public void reduce(Text key, Iterable<IntWritable> values, Context context)
```

```
            throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
```

```
        FileInputFormat.addInputPath(job, new Path(args[0]));

        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);

    }

}
```

Output:



```
2025-10-29 19:33:06 INFO  mapreduce.Job: Counters: 23
        File System Counters
                FILE: Number of bytes read=105
                FILE: Number of bytes written=290431
        Map-Reduce Framework
                Map input records=1
                Map output records=5
                Reduce input groups=4
                Reduce output records=4
                Combine input records=5
                Combine output records=4
Job Finished in 1.657 seconds
Output directory: /output

C:\hadoop_project>hdfs dfs -cat /output/part-r-00000
Hadoop      2
MapReduce   1
example     1
tutorial    1
```

EXPERIMENT NO: 7

Aim: To install Google App Engine and develop a simple web application for deployment on the cloud.

Code:

HelloServlet.java

```
package com.example.helloworld;



import java.io.IOException;

import javax.servlet.http.*;



@SuppressWarnings("serial")

public class HelloServlet extends HttpServlet {

    @Override

    public void doGet(HttpServletRequest req, HttpServletResponse resp)

            throws IOException {

        resp.setContentType("text/html");

        resp.getWriter().println("<h1>Hello from Google App Engine (Java)!</h1>");

        resp.getWriter().println("<p>This is a simple Java web app deployed on the cloud.</p>");

    }

}
```

📁 **web.xml**

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
```

```
         version="3.1">
  <servlet>
    <servlet-name>HelloServlet</servlet-name>
    <servlet-class>com.example.helloworld.HelloServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>HelloServlet</servlet-name>
    <url-pattern>/hello</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
</web-app>
```

🗂 **appengine-web.xml**

```
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
  <application>my-java-gae</application>
  <version>1</version>
  <threadsafe>true</threadsafe>
  <runtime>java11</runtime>

</appengine-web-app>
```

🗂 **index.html**

```
<!DOCTYPE html>

<html>

<head>

  <title>Hello App Engine</title>

</head>

<body>

  <h2>Welcome to My Google App Engine Java Application</h2>

  <p><a href="/hello">Click here</a> to access the HelloServlet.</p>

</body>

</html>
```

Output:

```
C:\HelloGAE> mvn appengine:run
INFO: Running application on http://localhost:8080/
INFO: Press Ctrl+C to stop.

C:\HelloGAE> gcloud app deploy
Services to deploy:
descriptor: [C:\HelloGAE\src\main\webapp\WEB-INF\appengine-web.xml]
source: [C:\HelloGAE]
target project: [my-java-gae]
target url: [https://my-java-gae.uc.r.appspot.com]

Do you want to continue (Y/n)?  Y
Beginning deployment...
Deploying...done.
Deployed service [default] to [https://my-java-gae.uc.r.appspot.com]
```

EXPERIMENT NO: 8

Aim: To launch a web application using Google App Engine, ensuring it is accessible and functional online.

Code:

index.html

```html
<!DOCTYPE html>

<html>

<head>

    <title>Google App Engine Demo</title>

</head>

<body>

    <h1>Welcome to My First App Engine Application!</h1>

    <form action="/hello" method="get">

        <label>Enter your name:</label>

        <input type="text" name="user" required>

        <input type="submit" value="Say Hello">

    </form>

</body>

</html>
```

HelloServlet.java

```java
package com.example.demo;
```

```
import java.io.IOException;

import javax.servlet.http.*;


@SuppressWarnings("serial")

public class HelloServlet extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse resp)

            throws IOException {

        String name = req.getParameter("user");

        if (name == null || name.trim().isEmpty()) {

            name = "Guest";

        }

        resp.setContentType("text/html");

        resp.getWriter().println("<h2>Hello, " + name + "!</h2>");

        resp.getWriter().println("<p>Your App Engine app is up and running!</p>");

    }

}
```

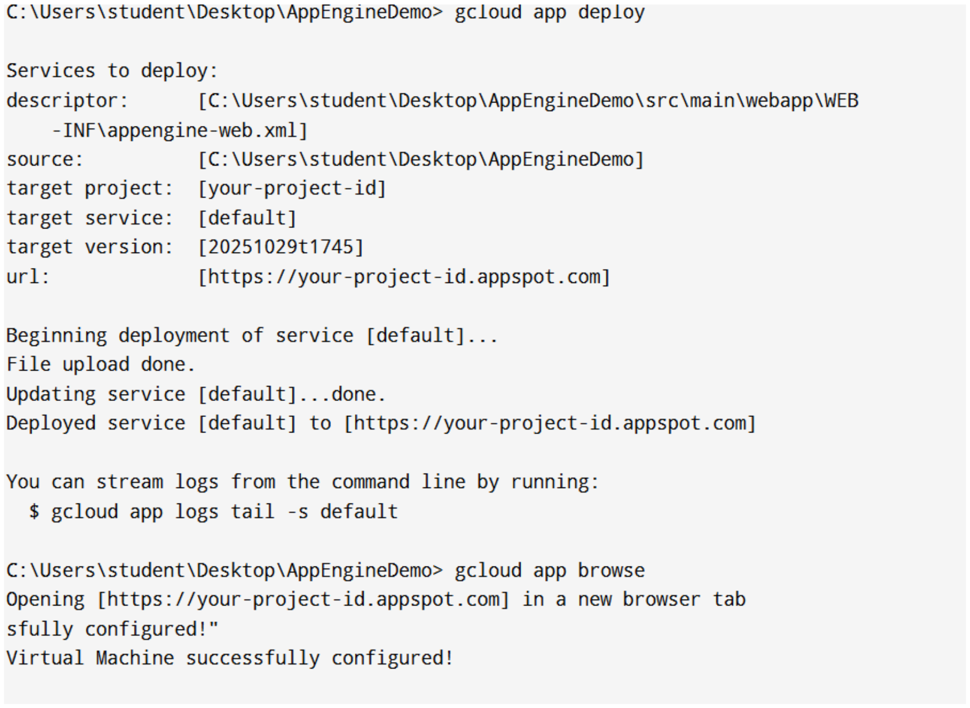
appengine-web.xml

```
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">

    <application>your-project-id</application>
    <version>1</version>

    <threadsafe>true</threadsafe>

    <runtime>java11</runtime>

</appengine-web-app>
```

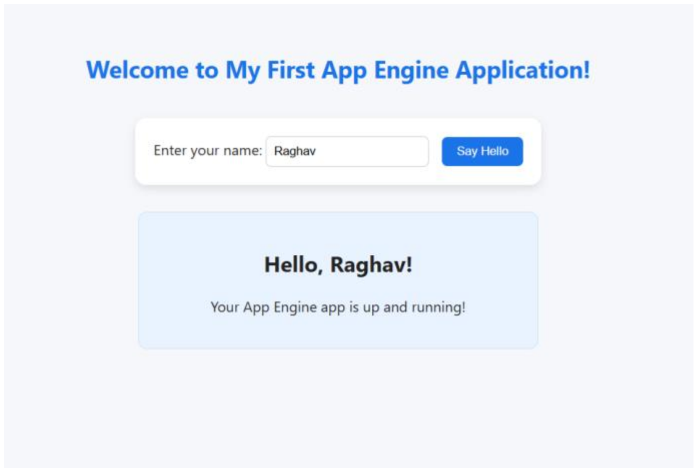
web.xml

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee

         http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"

         version="3.1">



    <servlet>

        <servlet-name>HelloServlet</servlet-name>

        <servlet-class>com.example.demo.HelloServlet</servlet-class>

    </servlet>



    <servlet-mapping>

        <servlet-name>HelloServlet</servlet-name>

        <url-pattern>/hello</url-pattern>
    </servlet-mapping>

</web-app>
```

Output:



```
C:\Users\student\Desktop\AppEngineDemo> gcloud app deploy

Services to deploy:
descriptor:      [C:\Users\student\Desktop\AppEngineDemo\src\main\webapp\WEB
    -INF\appengine-web.xml]
source:          [C:\Users\student\Desktop\AppEngineDemo]
target project:  [your-project-id]
target service:  [default]
target version:  [20251029t1745]
url:             [https://your-project-id.appspot.com]

Beginning deployment of service [default]...
File upload done.
Updating service [default]...done.
Deployed service [default] to [https://your-project-id.appspot.com]

You can stream logs from the command line by running:
  $ gcloud app logs tail -s default

C:\Users\student\Desktop\AppEngineDemo> gcloud app browse
Opening [https://your-project-id.appspot.com] in a new browser tab
sfully configured!"
Virtual Machine successfully configured!
```



Welcome to My First App Engine Application!

Enter your name: Raghav

Hello, Raghav!

Your App Engine app is up and running!

EXPERIMENT NO: 9

Aim: To install VirtualBox or VMware Workstation and set up various Linux distributions on a Windows host.

Course Outcome: CO4

Software Used: VirtualBox/VMware Workstation, Linux ISO images

Theory:

Virtualization technology allows multiple operating systems to run on a single physical machine by creating virtual machines (VMs). This is especially useful for testing, development, and educational purposes. VirtualBox and VMware Workstation are popular virtualization platforms that enable users to create and manage VMs with different operating systems, including various Linux distributions.

This setup allows users to experiment with different OS configurations, test software in different environments, and isolate projects for security and stability.

Procedure:

1. Download VirtualBox/VMware Workstation:
   - Visit the official websites and download the installers for VirtualBox or VMware Workstation.

2. Install Virtualization Software:
   - Run the installer and follow the on-screen instructions to install the chosen virtualization software on your Windows host.
   - Ensure that hardware virtualization is enabled in the BIOS settings of your computer.

3. Download Linux ISO Images:
   - Download the ISO images for the desired Linux distributions from their official websites (e.g., Ubuntu, CentOS, Fedora).

4. Create Virtual Machines:

- Open VirtualBox or VMware Workstation and create a new virtual machine for each Linux distribution.
- Allocate appropriate resources such as CPU, RAM, and disk space based on the requirements of the Linux distribution.

5. Install Linux Operating Systems:
   - Boot each VM with the corresponding Linux ISO image.
   - Follow the installation instructions for the Linux distribution, including setting up partitions, creating user accounts, and configuring system settings.

6. Post-Installation Configuration:
   - Install additional software and tools as needed.
   - Configure network settings to enable internet access and communication with the host machine.
   - Set up shared folders for easy file transfer between the host and VMs.

7. Testing and Usage:
   - Test the VMs to ensure they are functioning correctly.
   - Use the VMs to explore different Linux environments, develop software, or test applications.

Outputs:

```
Welcome to Ubuntu 22.04 LTS
---------------------------
Login: student
Password: ********

student@ubuntu:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04 LTS
Release:        22.04
Codename:       jammy

student@ubuntu:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.101  netmask 255.255.255.0  broadcast 192.168.56.255
        ether 08:00:27:3a:bb:11  txqueuelen 1000  (Ethernet)

student@ubuntu:~$ echo "Virtual Machine successfully configured!"
Virtual Machine successfully configured!
```

EXPERIMENT NO: 10

Aim: To simulate a cloud computing scenario using CloudSim and implement a scheduling algorithm.

Code:

```
import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.core.CloudSim;
import java.util.*;

public class CloudSimExample {
   public static void main(String[] args) {
      try {
         // Step 1: Initialize CloudSim
         int numUsers = 1;   // number of cloud users
         Calendar calendar = Calendar.getInstance();
         boolean traceFlag = false;

         CloudSim.init(numUsers, calendar, traceFlag);

         // Step 2: Create Datacenter
         Datacenter datacenter0 = createDatacenter("Datacenter_0");

         // Step 3: Create Broker
         DatacenterBroker broker = new DatacenterBroker("Broker_0");

         // Step 4: Create Virtual Machines
         List<Vm> vmlist = new ArrayList<>();

         int vmid = 0;
         int mips = 1000;
         long size = 10000; // image size (MB)
```

```
        int ram = 512;     // vm memory (MB)
        long bw = 1000;
        int pesNumber = 1; // number of CPUs
        String vmm = "Xen"; // Virtual Machine Monitor

        Vm vm1 = new Vm(vmid, broker.getId(), mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());
        Vm vm2 = new Vm(vmid + 1, broker.getId(), mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());

        vmlist.add(vm1);
        vmlist.add(vm2);

        broker.submitVmList(vmlist);

        // Step 5: Create Cloudlets
        List<Cloudlet> cloudletList = new ArrayList<>();

        int id = 0;
        long length = 40000;
        long fileSize = 300;
        long outputSize = 300;
        UtilizationModel utilizationModel = new UtilizationModelFull();

        for (int i = 0; i < 4; i++) {
            Cloudlet cloudlet = new Cloudlet(id + i, length, pesNumber, fileSize, outputSize,
utilizationModel, utilizationModel, utilizationModel);
            cloudlet.setUserId(broker.getId());
```

```
                cloudletList.add(cloudlet);
            }

            broker.submitCloudletList(cloudletList);

            // Step 6: Start Simulation
            CloudSim.startSimulation();

            // Step 7: Stop Simulation
            CloudSim.stopSimulation();

            // Step 8: Print results
            List<Cloudlet> newList = broker.getCloudletReceivedList();
            printCloudletList(newList);

        } catch (Exception e) {
            e.printStackTrace();
            System.out.println("Simulation terminated due to an error.");
        }
    }

    private static Datacenter createDatacenter(String name) {
        // Create a list of processing elements (PEs)
        List<Pe> peList = new ArrayList<>();
        int mips = 1000;
        peList.add(new Pe(0, new PeProvisionerSimple(mips))); // 1 CPU
        // Create a host with its resources
        int hostId = 0;
        int ram = 2048; // host memory (MB)
        long storage = 1000000; // host storage
        int bw = 10000;

        List<Host> hostList = new ArrayList<>();
        hostList.add(new Host(hostId, new RamProvisionerSimple(ram),
            new BwProvisionerSimple(bw), storage, peList,
            new VmSchedulerTimeShared(peList)));

        // Datacenter characteristics
        String arch = "x86"; // system architecture
        String os = "Linux"; // operating system
        String vmm = "Xen";
        double time_zone = 10.0;
        double cost = 3.0; // per second
        double costPerMem = 0.05; // per MB
        double costPerStorage = 0.1; // per MB
        double costPerBw = 0.1; // per MB

        DatacenterCharacteristics characteristics = new DatacenterCharacteristics(
            arch, os, vmm, hostList, time_zone, cost, costPerMem, costPerStorage, costPerBw);

        try {
          return new Datacenter(name, characteristics, new VmAllocationPolicySimple(hostList),
    new LinkedList<Storage>(), 0);
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }


    private static void printCloudletList(List<Cloudlet> list) {
        String indent = "    ";
        System.out.println("\n========== OUTPUT ==========");
        System.out.println("Cloudlet ID" + indent + "STATUS" + indent + "DataCenter ID" +
indent + "VM ID" + indent + "Time");
        System.out.println("-----------------------------------------------------------");


        for (Cloudlet cloudlet : list) {
            System.out.printf("%-10d %-10s %-15d %-10d %-10.2f%n",
                    cloudlet.getCloudletId(),
                    cloudlet.getStatusString(),
                    cloudlet.getResourceId(),
                    cloudlet.getVmId(),
                    cloudlet.getActualCPUTime());
        }
    }
}
```

Outputs:

```
Starting CloudSim simulation...

========== OUTPUT ==========
Cloudlet ID    STATUS       DataCenter ID   VM ID      Time
-----------------------------------------------------------
0              SUCCESS      2               0          40.00
1              SUCCESS      2               0          40.00
2              SUCCESS      2               1          40.00
3              SUCCESS      2               1          40.00

CloudSim simulation finished successfully.
```