External Practical

AIM:

Write a program in cloudsim using NetBeans IDE to create a six datacenters with six hosts and run cloudlets of six users on them.

CODE:

```
package org.cloudbus.cloudsim.examples;

import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;

import org.cloudbus.cloudsim.Cloudlet;
import
org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import
org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
import org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import
org.cloudbus.cloudsim.VmAllocationPolicySimple;
import
org.cloudbus.cloudsim.VmSchedulerSpaceShared;
import
org.cloudbus.cloudsim.VmSchedulerTimeShared;
import org.cloudbus.cloudsim.core.CloudSim;
import
org.cloudbus.cloudsim.provisioners.BwProvisionerSi
mple;
import
org.cloudbus.cloudsim.provisioners.PeProvisionerSim
ple;
import
org.cloudbus.cloudsim.provisioners.RamProvisionerSi
mple;
```

```
public class 18DCS007{

/** The cloudlet list. */
private static List<Cloudlet> cloudletList1;
private static List<Cloudlet> cloudletList2;
	private static List<Cloudlet> cloudletList3;
	private static List<Cloudlet> cloudletList4;
	private static List<Cloudlet> cloudletList5;
	private static List<Cloudlet> cloudletList6;

/** The vmlist. */
private static List<Vm> vmlist1;
private static List<Vm> vmlist2;
	private static List<Vm> vmlist3;
	private static List<Vm> vmlist4;
	private static List<Vm> vmlist5;
	private static List<Vm> vmlist6;

/**
 * Creates main() to run this example
 */
public static void main(String[] args) {

Log.printLine("Starting CloudSimExample4...");

try {
	// First step: Initialize the CloudSim package. It
should be called
	// before creating any entities.
	int num_user = 6;   // number of cloud users
	Calendar calendar = Calendar.getInstance();
	boolean trace_flag = false;  // mean trace events

	// Initialize the GridSim library
	CloudSim.init(num_user, calendar, trace_flag);

	// Second step: Create Datacenters
	//Datacenters are the resource providers in
CloudSim. We need at list one of them to run a
CloudSim simulation
	@SuppressWarnings("unused")
	Datacenter datacenter0 =
createDatacenter("Datacenter_0");
	@SuppressWarnings("unused")
```

```
        Datacenter datacenter1 =
createDatacenter("Datacenter_1");
                    @SuppressWarnings("unused")
        Datacenter datacenter2 =
createDatacenter("Datacenter_2");
                @SuppressWarnings("unused")
        Datacenter datacenter3 =
createDatacenter("Datacenter_3");
                @SuppressWarnings("unused")
        Datacenter datacenter4 =
createDatacenter("Datacenter_4");
                    @SuppressWarnings("unused")
        Datacenter datacenter5 =
createDatacenter("Datacenter_5");

        //Third step: Create Brokers
        DatacenterBroker broker1 = createBroker();
        int brokerId1 = broker1.getId();

        DatacenterBroker broker2 = createBroker();
        int brokerId2 = broker2.getId();

                //Third step: Create Brokers
        DatacenterBroker broker3 = createBroker();
        int brokerId3 = broker3.getId();

        DatacenterBroker broker4 = createBroker();
        int brokerId4 = broker4.getId();
                //Third step: Create Brokers
        DatacenterBroker broker5 = createBroker();
        int brokerId5 = broker5.getId();

                DatacenterBroker broker6 =
createBroker();
        int brokerId6 = broker6.getId();

        //Fourth step: Create one virtual machine
        vmlist1 = new ArrayList<Vm>();
        vmlist2 = new ArrayList<Vm>();
                vmlist3 = new ArrayList<Vm>();
                vmlist4 = new ArrayList<Vm>();
                vmlist5 = new ArrayList<Vm>();
                vmlist6 = new ArrayList<Vm>();

        //VM description
        int vmid = 0;
```

```
        int mips = 250;
        long size = 10000; //image size (MB)
        int ram = 512; //vm memory (MB)
        long bw = 1000;
        int pesNumber = 1; //number of cpus
        String vmm = "Xen"; //VMM name

        //create two VMs
        Vm vm1 = new Vm(vmid, brokerId1, mips,
pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());

        vmid++;
        Vm vm2 = new Vm(vmid, brokerId2, mips,
pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());

                vmid++;
        Vm vm3 = new Vm(vmid, brokerId3, mips,
pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());

                vmid++;
        Vm vm4 = new Vm(vmid, brokerId4, mips,
pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());

                vmid++;
        Vm vm5 = new Vm(vmid, brokerId5, mips,
pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());


                vmid++;
        Vm vm6 = new Vm(vmid, brokerId6, mips,
pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());

                //add the VMs to the vmList
        vmlist1.add(vm1);
        vmlist2.add(vm2);
                vmlist3.add(vm3);
                vmlist4.add(vm4);
                vmlist5.add(vm5);
                vmlist6.add(vm6);
```

```
        //submit vm list to the broker
        broker1.submitVmList(vmlist1);
                broker2.submitVmList(vmlist2);
                broker3.submitVmList(vmlist3);
                broker4.submitVmList(vmlist4);
                broker5.submitVmList(vmlist5);
                broker6.submitVmList(vmlist6);


        //Fifth step: Create two Cloudlets
        cloudletList1 = new ArrayList<Cloudlet>();
                cloudletList2 = new
ArrayList<Cloudlet>();
                cloudletList3 = new
ArrayList<Cloudlet>();
                cloudletList4 = new
ArrayList<Cloudlet>();
                cloudletList5 = new
ArrayList<Cloudlet>();
                cloudletList6 = new
ArrayList<Cloudlet>();

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

        Cloudlet cloudlet1 = new Cloudlet(id, length,
pesNumber, fileSize, outputSize, utilizationModel,
utilizationModel, utilizationModel);
        cloudlet1.setUserId(brokerId1);

        id++;
        Cloudlet cloudlet2 = new Cloudlet(id, length,
pesNumber, fileSize, outputSize, utilizationModel,
utilizationModel, utilizationModel);
        cloudlet2.setUserId(brokerId2);

                id++;
        Cloudlet cloudlet3 = new Cloudlet(id, length,
pesNumber, fileSize, outputSize, utilizationModel,
utilizationModel, utilizationModel);
```

```
        cloudlet3.setUserId(brokerId3);

                id++;
        Cloudlet cloudlet4 = new Cloudlet(id, length,
pesNumber, fileSize, outputSize, utilizationModel,
utilizationModel, utilizationModel);
        cloudlet4.setUserId(brokerId4);

                id++;
        Cloudlet cloudlet5 = new Cloudlet(id, length,
pesNumber, fileSize, outputSize, utilizationModel,
utilizationModel, utilizationModel);
        cloudlet5.setUserId(brokerId5);

                 id++;
        Cloudlet cloudlet6 = new Cloudlet(id, length,
pesNumber, fileSize, outputSize, utilizationModel,
utilizationModel, utilizationModel);
        cloudlet6.setUserId(brokerId6);

        //add the cloudlets to the list
        cloudletList1.add(cloudlet1);
        cloudletList2.add(cloudlet2);
                cloudletList3.add(cloudlet3);
                cloudletList4.add(cloudlet4);
                cloudletList5.add(cloudlet5);
                cloudletList6.add(cloudlet6);

        //submit cloudlet list to the broker
        broker1.submitCloudletList(cloudletList1);

broker2.submitCloudletList(cloudletList2);

broker3.submitCloudletList(cloudletList3);

broker4.submitCloudletList(cloudletList4);

broker5.submitCloudletList(cloudletList5);

broker6.submitCloudletList(cloudletList6);


        //bind the cloudlets to the vms. This way, the
broker
        // will submit the bound cloudlets only to the
specific VM
```

```
broker1.bindCloudletToVm(cloudlet1.getCloudletId(),
vm1.getId());

broker2.bindCloudletToVm(cloudlet2.getCloudletId(),
vm2.getId());

broker3.bindCloudletToVm(cloudlet3.getCloudletId(),
vm3.getId());

broker4.bindCloudletToVm(cloudlet4.getCloudletId(),
vm4.getId());

broker5.bindCloudletToVm(cloudlet5.getCloudletId(),
vm5.getId());

broker6.bindCloudletToVm(cloudlet6.getCloudletId(),
vm6.getId());

        // Sixth step: Starts the simulation
        CloudSim.startSimulation();


        // Final step: Print results when simulation is
over
        List<Cloudlet> newList1 =
broker1.getCloudletReceivedList();
                List<Cloudlet> newList2 =
broker2.getCloudletReceivedList();
                List<Cloudlet> newList3 =
broker3.getCloudletReceivedList();
                List<Cloudlet> newList4 =
broker4.getCloudletReceivedList();
                List<Cloudlet> newList5 =
broker5.getCloudletReceivedList();
                List<Cloudlet> newList6 =
broker6.getCloudletReceivedList();

        CloudSim.stopSimulation();

        printCloudletList(newList1);
           printCloudletList(newList2);
           printCloudletList(newList3);
           printCloudletList(newList4);
           printCloudletList(newList5);
           printCloudletList(newList6);
                Log.printLine("");
```

```
                Log.printLine("18DCS007 | RUDRA
BARAD");
                Log.printLine("");
        Log.printLine("CloudSimExample4
finished!");
}
catch (Exception e) {
        e.printStackTrace();
        Log.printLine("The simulation has been
terminated due to an unexpected error");
}


}

private static Datacenter createDatacenter(String
name){

// Here are the steps needed to create a
PowerDatacenter:
// 1. We need to create a list to store
//    our machine
List<Host> hostList = new ArrayList<Host>();

// 2. A Machine contains one or more PEs or
CPUs/Cores.
// In this example, it will have only one core.
List<Pe> peList = new ArrayList<Pe>();

int mips = 1000;

// 3. Create PEs and add these into a list.
peList.add(new Pe(0, new
PeProvisionerSimple(mips))); // need to store Pe id
and MIPS Rating

//4. Create Host with its id and list of PEs and add
them to the list of machines
int hostId=0;
int ram = 2048; //host memory (MB)
long storage = 1000000; //host storage
int bw = 10000;


//in this example, the VMAllocatonPolicy in use is
SpaceShared. It means that only one VM
```

```
//is allowed to run on each Pe. As each Host has only
one Pe, only one VM can run on each Host.
hostList.add(
                        new Host(
                                hostId,
                                new
RamProvisionerSimple(ram),
                                new
BwProvisionerSimple(bw),
                                storage,
                                peList,
                                new
VmSchedulerSpaceShared(peList)
                        )
                ); // This is our first machine



// 5. Create a DatacenterCharacteristics object that
stores the
//    properties of a data center: architecture, OS, list of
//    Machines, allocation policy: time- or space-shared,
time zone
//    and its price (G$/Pe time unit).
String arch = "x86";      // system architecture
String os = "Linux";          // operating system
String vmm = "Xen";
double time_zone = 10.0;         // time zone this
resource located
double cost = 3.0;              // the cost of using
processing in this resource
double costPerMem = 0.05;               // the cost of
using memory in this resource
double costPerStorage = 0.001;          // the cost of
using storage in this resource
double costPerBw = 0.0;                         // the cost
of using bw in this resource
LinkedList<Storage> storageList = new
LinkedList<Storage>();          //we are not adding SAN
devices by now

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

```
// 6. Finally, we need to create a PowerDatacenter
object.
Datacenter datacenter = null;
try {
        datacenter = new Datacenter(name,
characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
} catch (Exception e) {
        e.printStackTrace();
}

return datacenter;
}

//We strongly encourage users to develop their own
broker policies, to submit vms and cloudlets according
//to the specific rules of the simulated scenario
private static DatacenterBroker createBroker(){

DatacenterBroker broker = null;
try {
        broker = new DatacenterBroker("Broker");
} catch (Exception e) {
        e.printStackTrace();
        return null;
}
return broker;
}

/**
 * Prints the Cloudlet objects
 * @param list  list of Cloudlets
 */
private static void printCloudletList(List<Cloudlet>
list) {
int size = list.size();
Cloudlet cloudlet;

String indent = "    ";
Log.printLine();
Log.printLine("========== OUTPUT
==========");
Log.printLine("Cloudlet ID" + indent + "STATUS" +
indent +
```

```
                "Data center ID" + indent + "VM ID" +
indent + "Time" + indent + "Start Time" + indent +
"Finish Time");

DecimalFormat dft = new DecimalFormat("###.##");
for (int i = 0; i < size; i++) {
        cloudlet = list.get(i);
        Log.print(indent + cloudlet.getCloudletId() +
indent + indent);

        if (cloudlet.getCloudletStatus() ==
Cloudlet.SUCCESS){
                Log.print("SUCCESS");

                Log.printLine( indent + indent +
cloudlet.getResourceId() + indent + indent + indent +
cloudlet.getVmId() +
                                indent + indent +
dft.format(cloudlet.getActualCPUTime()) + indent +
indent + dft.format(cloudlet.getExecStartTime())+
                                indent + indent +
dft.format(cloudlet.getFinishTime()));
        }
}


}

}
```

OUTPUT

```
run:
Starting CloudSimExample4...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Datacenter_1 is starting...
Datacenter_2 is starting...
Datacenter_3 is starting...
Datacenter_4 is starting...
Datacenter_5 is starting...
Broker is starting...
Broker is starting...
Broker is starting...
Broker is starting...
Broker is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 6 resource(s)
0.0: Broker: Cloud Resource List received with 6 resource(s)
0.0: Broker: Cloud Resource List received with 6 resource(s)
0.0: Broker: Cloud Resource List received with 6 resource(s)
0.0: Broker: Cloud Resource List received with 6 resource(s)
0.0: Broker: Cloud Resource List received with 6 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.0: Broker: Trying to Create VM #2 in Datacenter_0
0.0: Broker: Trying to Create VM #3 in Datacenter_0
0.0: Broker: Trying to Create VM #4 in Datacenter_0
0.0: Broker: Trying to Create VM #5 in Datacenter_0
[VmScheduler.vmCreate] Allocation of VM #1 to Host #0 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #2 to Host #0 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #3 to Host #0 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #4 to Host #0 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #5 to Host #0 failed by MIPS
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Creation of VM #1 failed in Datacenter #2
0.1: Broker: Trying to Create VM #1 in Datacenter_1
0.1: Broker: Creation of VM #2 failed in Datacenter #2
0.1: Broker: Trying to Create VM #2 in Datacenter_1
0.1: Broker: Creation of VM #3 failed in Datacenter #2
0.1: Broker: Trying to Create VM #3 in Datacenter_1
0.1: Broker: Creation of VM #4 failed in Datacenter #2
0.1: Broker: Trying to Create VM #4 in Datacenter_1
0.1: Broker: Creation of VM #5 failed in Datacenter #2
0.1: Broker: Trying to Create VM #5 in Datacenter_1
[VmScheduler.vmCreate] Allocation of VM #2 to Host #0 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #3 to Host #0 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #4 to Host #0 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #5 to Host #0 failed by MIPS
```

```
0.2: Broker: Trying to Create VM #5 in Datacenter_2
[VmScheduler.vmCreate] Allocation of VM #3 to Host #0 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #4 to Host #0 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #5 to Host #0 failed by MIPS
0.30000000000000004: Broker: VM #2 has been created in Datacenter #4, Host #0
0.30000000000000004: Broker: Sending cloudlet 2 to VM #2
0.30000000000000004: Broker: Creation of VM #3 failed in Datacenter #4
0.30000000000000004: Broker: Trying to Create VM #3 in Datacenter_3
0.30000000000000004: Broker: Creation of VM #4 failed in Datacenter #4
0.30000000000000004: Broker: Trying to Create VM #4 in Datacenter_3
0.30000000000000004: Broker: Creation of VM #5 failed in Datacenter #4
0.30000000000000004: Broker: Trying to Create VM #5 in Datacenter_3
[VmScheduler.vmCreate] Allocation of VM #4 to Host #0 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #5 to Host #0 failed by MIPS
0.4: Broker: VM #3 has been created in Datacenter #5, Host #0
0.4: Broker: Sending cloudlet 3 to VM #3
0.4: Broker: Creation of VM #4 failed in Datacenter #5
0.4: Broker: Trying to Create VM #4 in Datacenter_4
0.4: Broker: Creation of VM #5 failed in Datacenter #5
0.4: Broker: Trying to Create VM #5 in Datacenter_4
[VmScheduler.vmCreate] Allocation of VM #5 to Host #0 failed by MIPS
0.5: Broker: VM #4 has been created in Datacenter #6, Host #0
0.5: Broker: Sending cloudlet 4 to VM #4
0.5: Broker: Creation of VM #5 failed in Datacenter #6
0.5: Broker: Trying to Create VM #5 in Datacenter_5
0.6: Broker: VM #5 has been created in Datacenter #7, Host #0
0.6: Broker: Sending cloudlet 5 to VM #5
160.1: Broker: Cloudlet 0 received
160.1: Broker: All Cloudlets executed. Finishing...
160.1: Broker: Destroying VM #0
Broker is shutting down...
160.2: Broker: Cloudlet 1 received
160.2: Broker: All Cloudlets executed. Finishing...
160.2: Broker: Destroying VM #1
Broker is shutting down...
160.3: Broker: Cloudlet 2 received
160.3: Broker: All Cloudlets executed. Finishing...
160.3: Broker: Destroying VM #2
Broker is shutting down...
160.4: Broker: Cloudlet 3 received
160.4: Broker: All Cloudlets executed. Finishing...
160.4: Broker: Destroying VM #3
Broker is shutting down...
160.5: Broker: Cloudlet 4 received
160.5: Broker: All Cloudlets executed. Finishing...
160.5: Broker: Destroying VM #4
Broker is shutting down...
160.6: Broker: Cloudlet 5 received
160.6: Broker: All Cloudlets executed. Finishing...
160.6: Broker: Destroying VM #5
```

```
160.6: Broker: All Cloudlets executed. Finishing...
160.6: Broker: Destroying VM #5
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Datacenter_1 is shutting down...
Datacenter_2 is shutting down...
Datacenter_3 is shutting down...
Datacenter_4 is shutting down...
Datacenter_5 is shutting down...
Broker is shutting down...
Broker is shutting down...
Broker is shutting down...
Broker is shutting down...
Broker is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0        SUCCESS        2            0        160        0.1        160.1

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    1        SUCCESS        3            1        160        0.2        160.2

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    2        SUCCESS        4            2        160        0.3        160.3

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    3        SUCCESS        5            3        160        0.4        160.4

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    4        SUCCESS        6            4        160        0.5        160.5

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    5        SUCCESS        7            5        160        0.6        160.6

18DCS007 | RUDRA BARAD

CloudSimExample4 finished!
[0x7FFF38A16970] ANOMALY: meaningless REX prefix used
BUILD SUCCESSFUL (total time: 0 seconds)
```

CONCLUSION

Successfully completed the given practical.