**PseudoCode of problem 9**

**public** **class problem 8**

{

//Creating two virtual machines in private class of Vm

Vm[] vm = **new** Vm[1];

**for**(**int** i=0;i<1;i++){

vm[i] = **new** Vm(idShift + i, userId, mips, pesNumber, ram, bw, size, vmm, **new** CloudletSchedulerTimeShared());

list.add(vm[i]);

**}**

//creating 1 cloudlets in private class of cloudlets

//Making Cloudlets with different properties(create a loop and divide the cloudlets and change the properties) 20 50 and 100 cloudlets;

**for**(**int** i=0;i<cloudlets/4;i++){

**long** length = 40000;

**long** fileSize = 300;

**long** outputSize = 300;

**int** pesNumber = 1;

UtilizationModel utilizationModel = **new** UtilizationModelFull();

Cloudlet[] cloudlet = **new** Cloudlet[cloudlets];

cloudlet[i] = **new** Cloudlet(idShift + i, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet[i].setUserId(userId);

list.add(cloudlet[i]);

}

**for**(**int** i=cloudlets/4;i<cloudlets/3;i++){

**long** length = 400000;

**long** fileSize = 3000;

**long** outputSize = 3000;

**int** pesNumber = 1;

UtilizationModel utilizationModel = **new** UtilizationModelFull();

Cloudlet[] cloudlet = **new** Cloudlet[cloudlets];

cloudlet[i] = **new** Cloudlet(idShift + i, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet[i].setUserId(userId);

list.add(cloudlet[i]);

}

**for**(**int** i=cloudlets/3;i<cloudlets/2;i++){

**long** length = 4000000;

**long** fileSize = 30000;

**long** outputSize = 30000;

**int** pesNumber = 1;

UtilizationModel utilizationModel = **new** UtilizationModelFull();

Cloudlet[] cloudlet = **new** Cloudlet[cloudlets];

cloudlet[i] = **new** Cloudlet(idShift + i, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet[i].setUserId(userId);

list.add(cloudlet[i]);

}

**for**(**int** i=cloudlets/2;i<cloudlets;i++){

**long** length = 4000000;

**long** fileSize = 30000;

**long** outputSize = 30000;

**int** pesNumber = 1;

UtilizationModel utilizationModel = **new** UtilizationModelFull();

Cloudlet[] cloudlet = **new** Cloudlet[cloudlets];

cloudlet[i] = **new** Cloudlet(idShift + i, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet[i].setUserId(userId);

list.add(cloudlet[i]);

**}**

//load the network topology file

**NetworkTopology.*buildNetworkTopology*("C:\Users\vaibhav\Desktop\Cloud experiments 5 th sem\cloudsim 3.0.3\ bin\org\ cloudbus\cloudsim\ examples\network\topology.brite");**

**NetworkTopology.*mapNode*(datacenter0.getId(),briteNode);**

Main()

{

//creating data ceneter and datacentre broker

Datacenter datacenter0 = *createDatacenter*("Datacenter\_0");

DatacenterBroker broker = *createBroker*("Broker\_0");

**int** brokerId = broker.getId();

*vmlist* = *createVM*(brokerId, 1, 100); //creating 1 vms

*cloudletList* = *createCloudlet*(brokerId, 10, 100); // creating 2/5/10 cloudlets

steps needed to create a PowerDatacenter:

//private class of datacentre

{

List<Host> hostList = **new** ArrayList<Host>();

//add Pe list to data centers so both virtual machines gets the successful status

peList1.add(**new** Pe(0, **new** PeProvisionerSimple(mips)));

}

**private** **static** **void** printCloudletList

**{**

Log.*printLine*("Cloudlet ID" + indent + "STATUS" + indent +"Data center ID" + indent + "VM ID" + indent + indent + "Time" + indent + "Start Time" + indent + "Finish Time");

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

**}**