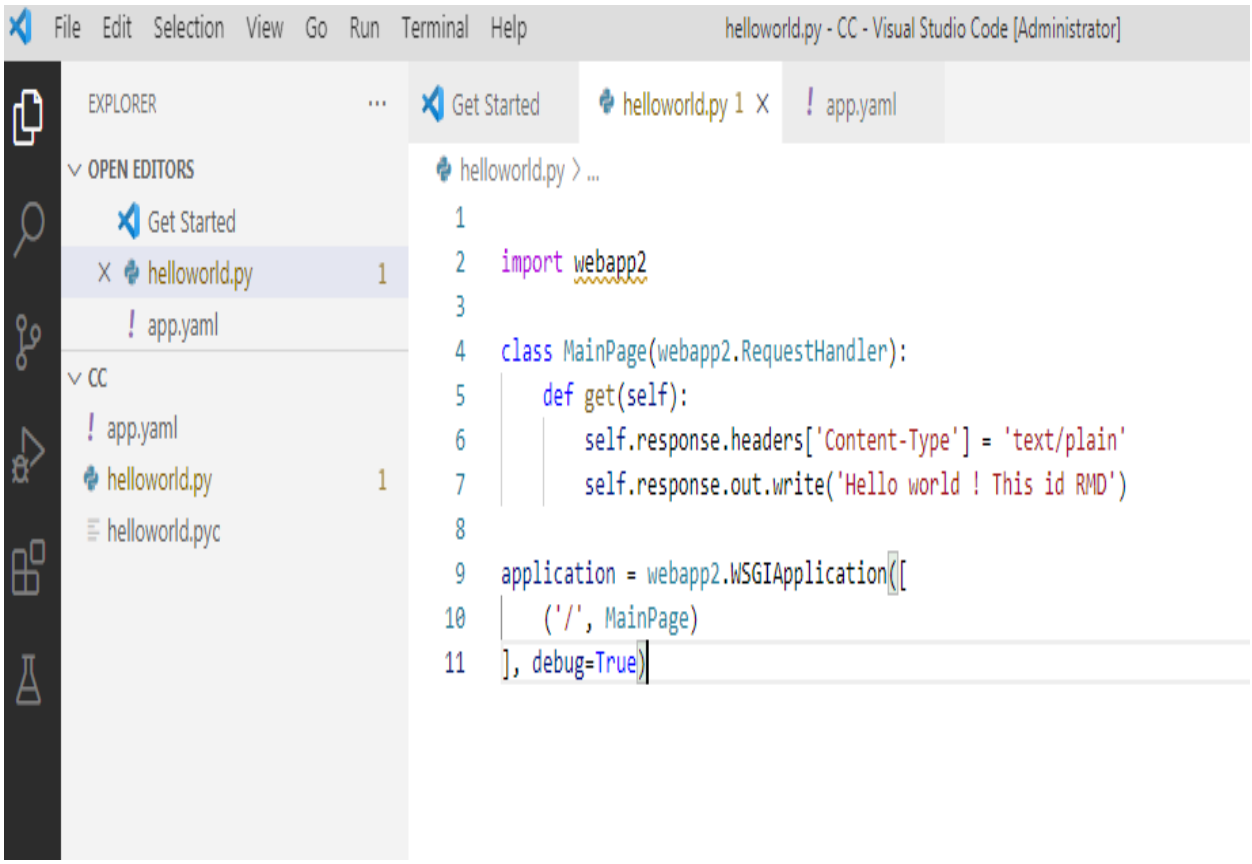
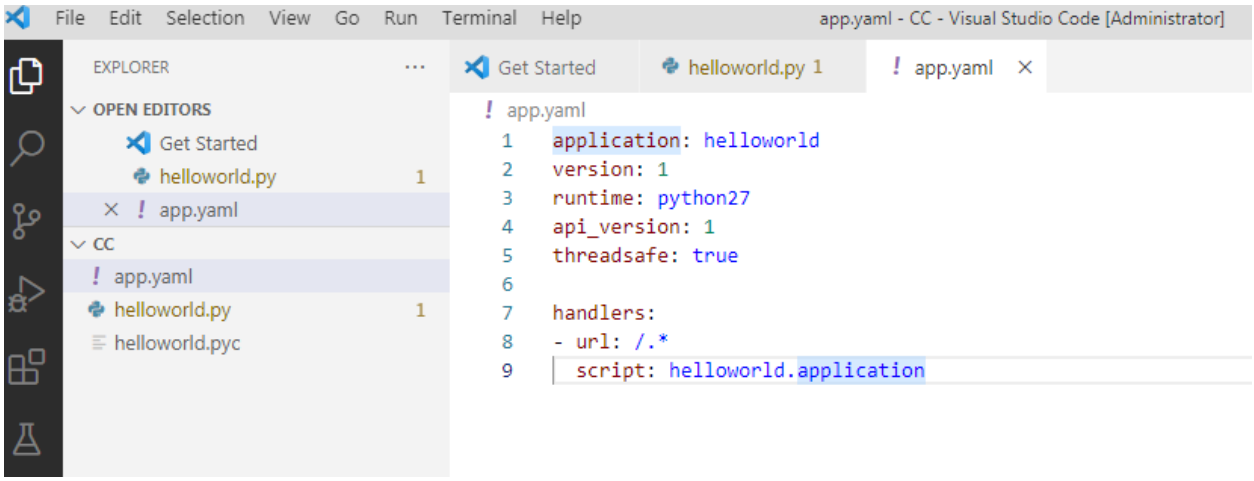
AssignmentNo 1

```
import webapp2

class MainPage(webapp2.RequestHandler):
    def get(self):
        self.response.headers['Content-Type'] = 'text/plain'
        self.response.out.write('Hello world ! This id RMD')

application = webapp2.WSGIApplication([
    ('/', MainPage)
], debug=True)
```




File Edit Selection View Go Run Terminal Help
app.yaml - CC - Visual Studio Code [Administrator]
EXPLORER
OPEN EDITORS
Get Started
helloworld.py 1
app.yaml
CC
app.yaml
helloworld.py 1
helloworld.pyc
Get Started
helloworld.py 1
app.yaml
app.yaml
1 application: helloworld
2 version: 1
3 runtime: python27
4 api_version: 1
5 threadsafe: true
6
7 handlers:
8 - url: /.*
9 script: helloworld.application

Assignment No 2

Main.py

```
import
os
        import json
        import urllib
        import webapp2
        from google.appengine.ext.webapp import template
        class MainPage(webapp2.RequestHandler):
            def get(self):
                template_values = {}
                path = os.path.join(os.path.dirname(__file__), 'index.html')
                self.response.out.write(template.render(path, template_values))
            def post(self):
                pincode = self.request.get('zipCode')
                if not pincode.isnumeric() or not len(pincode)==6:
                    template_values = {
                        "error" : "Incorrect Pin Code (String / False Code entered)"
                    }
                    path = os.path.join(os.path.dirname(__file__), 'index.html')
                    return self.response.out.write(template.render(path, template_values))
                url = "https://api.postalpincode.in/pincode/"+ pincode
                data = urllib.urlopen(url).read()
                data = json.loads(data)
                post_office = data[0]['PostOffice'][0]['State']
                name = data[0]['PostOffice'][0]['Name']
                block = data[0]['PostOffice'][0]['Block']
                district = data[0]['PostOffice'][0]['District']
                template_values = {
                    "post_office": post_office,
                    "name": name,
                    "block": block,
                    "district": district
                }
                path = os.path.join(os.path.dirname(__file__), 'results.html')
                self.response.out.write(template.render(path, template_values))


        app = webapp2.WSGIApplication([('/', MainPage)], debug=True)
```

App.yaml

```
runtime:
python27
           threadsafe: true
           handlers:
           - url: /
             script: main.app
```

Index.html

```
<html
>
             <style>
                 .weatherText {
                     font-family: "Lato", "sans-serif";
                     font-size: 24px;
                     text-align: center;
                 }
                 #weatherForm {
                     padding: 20px;
                 }
                 #weatherSubmit {
                     color: white;
                     background-color: #083375;
                     padding: 5px 20px;
                     border-radius: 5px;
                     margin-top: 20px;
                 }
                 #weatherSubmit:hover {
                     cursor: pointer;
                 }
                 body {
                     display: flex;
                     justify-content: center;
                     align-items: center;
                 }
```

```
        .card {
            border: 2px solid black;
            width: 50%;
            justify-content: center;
            align-items: center;
        }
    </style>
    <head>
        <title class="alignct">Post Office Finder</title>
        <link

href="https://fonts.googleapis.com/css2?family=Lato:wght@400;700&display=swap"
            rel="stylesheet"
        />
    </head>
    <body>
        <h1 id="error_head" style="display: none" value="{{error}}">
            {{error}}
        </h1>
        <div class="card">
            <h2 class="weatherText">Post Office Finder Using WebApp</h2>
            <form class="weatherText" id="weatherForm" action="/" method="post">
                Location Zip Code:
                <input
                    class="weatherText"
                    id="weatherInput"
                    type="text"
                    name="zipCode"
                /><br />
                <input
                    class="weatherText"
                    id="weatherSubmit"
                    type="submit"
                    value="Submit"
                />
            </form>
        </div>
        <script>
            let err = document.getElementById("error_head");
            function myFunction() {
                alert("Enter Valid Pin Code");
            }
            if (err) {
```

```
            myFunction();
        }
    </script>
</body>
</html>
```

Result.html

```
<!DOCTYP
E html>
        <html lang="en">
            <style>
                body {
                    display: flex;
                    justify-content: center;
                    align-item: center;
                }
                #weatherResults {
                    background-color: #83e9c2;
                    font-family: "Lato", sans-serif;
                    font-size: 24px;
                    padding: 30px;
                    display: inline-block;
                    text-align: center;
                    margin: 20px;
                    margin-top: 10%;
                    border: 2px solid black;
                    border-radius: 5px;
                }
            </style>
            <head>
                <meta charset="UTF-8" />
                <title>Post Office Information</title>
                <link

        href="https://fonts.googleapis.com/css2?family=Lato:wght@400;700&displa
        y=swap"
                    rel="stylesheet"
                />
            </head>
            <body>
```

```
        <div id="weatherResults">
            <table>
                <tr>
                    <th>
                        <h3>State of Post Office :</h3>
                    </th>
                    <th>
                        <h3>{{ post_office }}</h3>
                    </th>
                </tr>
                <tr>
                    <th>
                        <h3>Name of Post Office :</h3>
                    </th>
                    <th>
                        <h3>{{ name }}</h3>
                    </th>
                </tr>
                <tr>
                    <th>
                        <h3>Block of Post Office:</h3>
                    </th>
                    <th>
                        <h3>{{ block }}</h3>
                    </th>
                </tr>
                <tr>
                    <th>
                        <h3>District of Post Office:</h3>
                    </th>
                    <th>
                        <h3>{{ district }}</h3>
                    </th>
                </tr>
            </table>
        </div>
    </body>
</html>
```

ASSIGNMENT 03

```
</html>
```

Result.html

```
<!DOCTY
PE html>
                    <html lang="en">
                      <style>
                        body {
                          display: flex;
                          justify-content: center;
                          align-item: center;
                        }
                        #weatherResults {
                          background-color: #83e9c2;
                          font-family: "Lato", sans-serif;
                          font-size: 24px;
                          padding: 30px;
                          display: inline-block;
                          text-align: center;
                          margin: 20px;
                          margin-top: 10%;
                          border: 2px solid black;
                          border-radius: 5px;
                        }
                      </style>
                      <head>
                        <meta charset="UTF-8" />
                        <title>Post Office Information</title>
                        <link
                          href="https://fonts.googleapis.com/css2?family=Lato:wght@400;700&display=swap"
                          rel="stylesheet"
                        />
                      </head>
                      <body>
                        <div id="weatherResults">
                          <table>
                            <tr>
                              <th>
                                <h3>State of Post Office :</h3>
```

```
                </th>
                <th>
                    <h3>{{ post_office }}</h3>
                </th>
            </tr>
            <tr>
                <th>
                    <h3>Name of Post Office :</h3>
                </th>
                <th>
                    <h3>{{ name }}</h3>
                </th>
            </tr>
            <tr>
                <th>
                    <h3>Block of Post Office:</h3>
                </th>
                <th>
                    <h3>{{ block }}</h3>
                </th>
            </tr>
            <tr>
                <th>
                    <h3>District of Post Office:</h3>
                </th>
                <th>
                    <h3>{{ district }}</h3>
                </th>
            </tr>
        </table>
    </div>
</body>
</html>
```

Assignment No 3

Constants.java

```
package
<package_name>;

public class Constants {

    public static final int NO_OF_TASKS = 30; // number of Cloudlets;

    public static final int NO_OF_DATA_CENTERS = 5; // number of
Datacenters;

    public static final int POPULATION_SIZE = 25; // Number of Partic

}
```

DatacenterCreator.java

```
package
<package_name>;
import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
public class DatacenterCreator {
   public static Datacenter createDatacenter(String name) {
      // Here are the steps needed to create a PowerDatacenter:
      // 1. We need to create a list to store one or more Machines
```

```
List<Host> hostList = new ArrayList<Host>();
// 2. A Machine contains one or more PEs or CPUs/Cores. Therefore, should
//    create a list to store these PEs before creating a Machine.
List<Pe> peList = new ArrayList<Pe>();
int mips = 1000;
// 3. Create PEs and add these into the list.
peList.add(new Pe(0, new PeProvisionerSimple(mips)));
//4. Create Hosts 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;
hostList.add(
      new Host(
            hostId,
            new RamProvisionerSimple(ram),
            new BwProvisionerSimple(bw),
            storage,
            peList,
            new VmSchedulerTimeShared(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 = 0.1;    // the cost of using storage in this resource
double costPerBw = 0.1;            // 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;
        }
    }
```

GenerateMatrices.java

```
package
<package_name>;

        import java.io.*;

        public class GenerateMatrices {
            private static double[][] commMatrix, execMatrix;
            private File commFile = new File("CommunicationTimeMatrix.txt");
            private File execFile = new File("ExecutionTimeMatrix.txt");

            public GenerateMatrices() {
                commMatrix = new double[Constants.NO_OF_TASKS][Constants.NO_OF_DATA_CENTERS];
                execMatrix = new double[Constants.NO_OF_TASKS][Constants.NO_OF_DATA_CENTERS];
                try {
                    if (commFile.exists() && execFile.exists()) {
                        readCostMatrix();
                    } else {
                        initCostMatrix();
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
```

```
}

private void initCostMatrix() throws IOException {
    System.out.println("Initializing new Matrices...");
    BufferedWriter commBufferedWriter = new BufferedWriter(new FileWriter(commFile));
    BufferedWriter execBufferedWriter = new BufferedWriter(new FileWriter(execFile));

    for (int i = 0; i < Constants.NO_OF_TASKS; i++) {
        for (int j = 0; j < Constants.NO_OF_DATA_CENTERS; j++) {
            commMatrix[i][j] = Math.random() * 600 + 20;
            execMatrix[i][j] = Math.random() * 500 + 10;
            commBufferedWriter.write(String.valueOf(commMatrix[i][j]) + ' ');
            execBufferedWriter.write(String.valueOf(execMatrix[i][j]) + ' ');
        }
        commBufferedWriter.write('\n');
        execBufferedWriter.write('\n');
    }
    commBufferedWriter.close();
    execBufferedWriter.close();
}

private void readCostMatrix() throws IOException {
    System.out.println("Reading the Matrices...");
    BufferedReader commBufferedReader = new BufferedReader(new FileReader(commFile));

    int i = 0, j = 0;
    do {
        String line = commBufferedReader.readLine();
        for (String num : line.split(" ")) {
            commMatrix[i][j++] = new Double(num);
        }
        ++i;
        j = 0;
    } while (commBufferedReader.ready());


    BufferedReader execBufferedReader = new BufferedReader(new FileReader(execFile));

    i = j = 0;
```

```
                do {
                    String line = execBufferedReader.readLine();
                    for (String num : line.split(" ")) {
                        execMatrix[i][j++] = new Double(num);
                    }
                    ++i;
                    j = 0;
                } while (execBufferedReader.ready());
            }

            public static double[][] getCommMatrix() {
                return commMatrix;
            }

            public static double[][] getExecMatrix() {
                return execMatrix;
            }
        }
```

SJFDatacenterBroker.java

me>;

```
import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.core.CloudSimTags;
import org.cloudbus.cloudsim.core.SimEvent;

import java.util.ArrayList;
import java.util.List;

public class SJFDatacenterBroker extends DatacenterBroker {
```

```
    SJFDatacenterBroker(String name) throws Exception {
        super(name);
    }

    public void scheduleTaskstoVms() {
        int reqTasks = cloudletList.size();
        int reqVms = vmList.size();
        Vm vm = vmList.get(0);

        for (int i = 0; i < reqTasks; i++) {
            bindCloudletToVm(i, (i % reqVms));
m.out.println("Task" + cloudletList.get(i).getCloudletId() + " is bound with VM" + vmList.get(i % reqVms).getId());
        }

        //System.out.println("reqTasks: "+ reqTasks);

        ArrayList<Cloudlet> list = new ArrayList<Cloudlet>();
        for (Cloudlet cloudlet : getCloudletReceivedList()) {
            list.add(cloudlet);
        }

        //setCloudletReceivedList(null);

        Cloudlet[] list2 = list.toArray(new Cloudlet[list.size()]);

        //System.out.println("size :"+list.size());

        Cloudlet temp = null;

        int n = list.size();

        for (int i = 0; i < n; i++) {
            for (int j = 1; j < (n - i); j++) {
    if (list2[j - 1].getCloudletLength() / (vm.getMips() * vm.getNumberOfPes()) > list2[j].getCloudletLength() /
   (vm.getMips() * vm.getNumberOfPes())) {
                    //swap the elements!
                    //swap(list2[j-1], list2[j]);
                    temp = list2[j - 1];
                    list2[j - 1] = list2[j];
```

```
                list2[j] = temp;
            }
            // printNumbers(list2);
        }
    }

    ArrayList<Cloudlet> list3 = new ArrayList<Cloudlet>();

    for (int i = 0; i < list2.length; i++) {
        list3.add(list2[i]);
    }
    //printNumbers(list);

    setCloudletReceivedList(list);

    //System.out.println("\n\tSJFS Broker Schedules\n");
    //System.out.println("\n");
}

public void printNumber(Cloudlet[] list) {
    for (int i = 0; i < list.length; i++) {
        System.out.print(" " + list[i].getCloudletId());
        System.out.println(list[i].getCloudletStatusString());
    }
    System.out.println();
}

public void printNumbers(ArrayList<Cloudlet> list) {
    for (int i = 0; i < list.size(); i++) {
        System.out.print(" " + list.get(i).getCloudletId());
    }
    System.out.println();
}

@Override
protected void processCloudletReturn(SimEvent ev) {
    Cloudlet cloudlet = (Cloudlet) ev.getData();
    getCloudletReceivedList().add(cloudlet);
    Log.printLine(CloudSim.clock() + ": " + getName() + ": Cloudlet " + cloudlet.getCloudletId()
```

```
            + " received");
    cloudletsSubmitted--;
    if (getCloudletList().size() == 0 && cloudletsSubmitted == 0) {
        scheduleTaskstoVms();
        cloudletExecution(cloudlet);
    }
}

protected void cloudletExecution(Cloudlet cloudlet) {

    if (getCloudletList().size() == 0 && cloudletsSubmitted == 0) { // all cloudlets executed
        Log.printLine(CloudSim.clock() + ": " + getName() + ": All Cloudlets executed. Finishing...");
        clearDatacenters();
        finishExecution();
    } else { // some cloudlets haven't finished yet
        if (getCloudletList().size() > 0 && cloudletsSubmitted == 0) {
            // all the cloudlets sent finished. It means that some bount
            // cloudlet is waiting its VM be created
            clearDatacenters();
            createVmsInDatacenter(0);
        }
    }
}

@Override
protected void processResourceCharacteristics(SimEvent ev) {
    DatacenterCharacteristics characteristics = (DatacenterCharacteristics) ev.getData();
    getDatacenterCharacteristicsList().put(characteristics.getId(), characteristics);

    if (getDatacenterCharacteristicsList().size() == getDatacenterIdsList().size()) {
        distributeRequestsForNewVmsAcrossDatacenters();
    }
}

protected void distributeRequestsForNewVmsAcrossDatacenters() {
    int numberOfVmsAllocated = 0;
    int i = 0;

    final List<Integer> availableDatacenters = getDatacenterIdsList();
```

```
            for (Vm vm : getVmList()) {
                int datacenterId = availableDatacenters.get(i++ % availableDatacenters.size());
                String datacenterName = CloudSim.getEntityName(datacenterId);

                if (!getVmsToDatacentersMap().containsKey(vm.getId())) {
ıdSim.clock() + ": " + getName() + ": Trying to Create VM #" + vm.getId() + " in " + datacenterName);
                    sendNow(datacenterId, CloudSimTags.VM_CREATE_ACK, vm);
                    numberOfVmsAllocated++;
                }
            }

            setVmsRequested(numberOfVmsAllocated);
            setVmsAcks(0);
        }
```

SJF_Scheduler.java

e <

```
package_name>;
import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
//import utils.Constants;
//import utils.DatacenterCreator;
//import utils.GenerateMatrices;

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

public class SJF_Scheduler {
```

```
        private static List<Cloudlet> cloudletList;
        private static List<Vm> vmList;
        private static Datacenter[] datacenter;
        private static double[][] commMatrix;
        private static double[][] execMatrix;

        private static List<Vm> createVM(int userId, int vms) {
            //Creates a container to store VMs. This list is passed to the broker later
            LinkedList<Vm> list = new LinkedList<Vm>();

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

            //create VMs
            Vm[] vm = new Vm[vms];

            for (int i = 0; i < vms; i++) {
                vm[i] = new Vm(datacenter[i].getId(), userId, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerSpaceShared());
                list.add(vm[i]);
            }

            return list;
        }

        private static List<Cloudlet> createCloudlet(int userId, int cloudlets, int idShift) {
            // Creates a container to store Cloudlets
            LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();

            //cloudlet parameters
            long fileSize = 300;
            long outputSize = 300;
            int pesNumber = 1;
            UtilizationModel utilizationModel = new UtilizationModelFull();
```

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

    for (int i = 0; i < cloudlets; i++) {
        int dcId = (int) (Math.random() * Constants.NO_OF_DATA_CENTERS);
        long length = (long) (1e3 * (commMatrix[i][dcId] + execMatrix[i][dcId]));
        cloudlet[i] = new Cloudlet(idShift + i, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);
        // setting the owner of these Cloudlets
        cloudlet[i].setUserId(userId);
        cloudlet[i].setVmId(dcId + 2);
        list.add(cloudlet[i]);
    }
    return list;
}

public static void main(String[] args) {
    Log.printLine("Starting SJF Scheduler...");

    new GenerateMatrices();
    execMatrix = GenerateMatrices.getExecMatrix();
    commMatrix = GenerateMatrices.getCommMatrix();

    try {
        int num_user = 1;   // number of grid users
        Calendar calendar = Calendar.getInstance();
        boolean trace_flag = false;  // mean trace events

        CloudSim.init(num_user, calendar, trace_flag);

        // Second step: Create Datacenters
        datacenter = new Datacenter[Constants.NO_OF_DATA_CENTERS];
        for (int i = 0; i < Constants.NO_OF_DATA_CENTERS; i++) {
            datacenter[i] = DatacenterCreator.createDatacenter("Datacenter_" + i);
        }

        //Third step: Create Broker
        SJFDatacenterBroker broker = createBroker("Broker_0");
        int brokerId = broker.getId();

        //Fourth step: Create VMs and Cloudlets and send them to broker
```

```
            vmList = createVM(brokerId, Constants.NO_OF_DATA_CENTERS);
            cloudletList = createCloudlet(brokerId, Constants.NO_OF_TASKS, 0);

            broker.submitVmList(vmList);
            broker.submitCloudletList(cloudletList);

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

            // Final step: Print results when simulation is over
            List<Cloudlet> newList = broker.getCloudletReceivedList();
            //newList.addAll(globalBroker.getBroker().getCloudletReceivedList());

            CloudSim.stopSimulation();

            printCloudletList(newList);

            Log.printLine(SJF_Scheduler.class.getName() + " finished!");
        } catch (Exception e) {
            e.printStackTrace();
            Log.printLine("The simulation has been terminated due to an unexpected error");
        }
    }

    private static SJFDatacenterBroker createBroker(String name) throws Exception {
        return new SJFDatacenterBroker(name);
    }
    /**
     * 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 + indent + "Time" +
            indent + "Start Time" +
            indent + "Finish Time" +
            indent + "Waiting Time");

    DecimalFormat dft = new DecimalFormat("###.##");
dft.setMinimumIntegerDigits(2);
    for (int i = 0; i < size; i++) {
      cloudlet = list.get(i);
      Log.print(indent + dft.format(cloudlet.getCloudletId()) + indent + indent);

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

        Log.printLine(indent + indent + dft.format(cloudlet.getResourceId()) +
              indent + indent + indent + dft.format(cloudlet.getVmId()) +
              indent + indent + dft.format(cloudlet.getActualCPUTime()) +
              indent + indent + dft.format(cloudlet.getExecStartTime()) +
              indent + indent + indent + dft.format(cloudlet.getFinishTime())+
              indent + indent + indent + dft.format(cloudlet.getWaitingTime()));
      }
    }
    double makespan = calcMakespan(list);
    Log.printLine("Makespan using SJF: " + makespan);
  }

  private static double calcMakespan(List<Cloudlet> list) {
    double makespan = 0;
    double[] dcWorkingTime = new double[Constants.NO_OF_DATA_CENTERS];

    for (int i = 0; i < Constants.NO_OF_TASKS; i++) {
      int dcId = list.get(i).getVmId() % Constants.NO_OF_DATA_CENTERS;
      if (dcWorkingTime[dcId] != 0) --dcWorkingTime[dcId];
      dcWorkingTime[dcId] += execMatrix[i][dcId] + commMatrix[i][dcId];
      makespan = Math.max(makespan, dcWorkingTime[dcId]);
    }
```

```
        return makespan;
    }
}
```

Video Link for this assignment:

https://www.youtube.com/watch?v=6-2y3yA2w3M

Assignment No. 1

Title-

Install Google App Engine. Create hello world app and other simple web applications using python/java

Steps-

Download python from- https://www.python.org/downloads/

Download Google Cloud SDK from- https://cloud.google.com/sdk/docs/install#windows

Launch the installer and follow the prompts

Perform initial setup by running *gcloud init*

Grant authorization to Cloud SDK tools to access Google Cloud

- Write python file with hello world statement
- Write app.yaml configuration file
- Open the shell
- Run the application with the following command in shell:
- *cmd> py google-cloud-sdk\bin\dev_appserver.py <path to the directory where application reside>*
- Open the web browser and type *http://localhost:8080*

Assignment No. 2

- **Title-**
  - Use GAE launcher to launch the web applications

- **Steps-**
  - Already you have installed google cloud SDK and python
  - Write the configuration file
  - Write the web application file
  - Deploy and run it

Assignment No. 3

- **Title-**
  - Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim
- **Steps-**
  - Download the cloudsim jar file- https://github.com/Cloudslab/cloudsim/releases/tag/cloudsim-4.0
  - Download and install java sdk from- https://www.oracle.com/java/technologies/downloads/#java11
  - Download and install eclipse from- https://www.eclipse.org/downloads/packages/release/kepler/sr1/eclipse-ide-java-developers
  - Write the java files for the any task scheduling algorithm
  - Add cloudsim jar file as external jar file in the build configuration
  - Run application

- **Steps :**
- Install any of the IDE for running JAVA applications (eclipse recommended)
- Install JDK and JRE for the same
- Add the jdk\bin path to the environment variables Open environment variables window, add the following to the path variable
- Do include your bin path wherever you have installed JDK Mine is as following : > C:\Program Files\Java\jdk-14.0.1\bin
- Open eclipse in your confined workspace
- Click on new and open a new **JAVA Project**, give it a name
- Create a package inside he src folder.
- Dump in all the files here inside the package.
- Now right click on the project and choose **configure build path**.
- Click on the libraries section and add external jars
- Extract the Cloudsim.tar file you downloaded
- Now import the jar files in that Cloudsim.tar into the external jars.
- Do remeber to change the name of the package in all the source files.
- Now right click on the project and run the project as **JAVA Application**.
- Select the SJF_Scheduler.java file present in the list.

Assignment No. 4

- **Title-**
  - Find a procedure to transfer the files from one virtual machine to another virtual machine

- **Steps-**
  - Download and install Oracle's Virtual Box- https://www.virtualbox.org/wiki/Downloads
  - Download Ubuntu VMDK Image- https://app.vagrantup.com/bento/boxes/ubuntu-18.04
  - Launch Virtualbox and create a new VM

- Click on new and mention the Name and the machine folder along with the Type and Version of the Machine to be created.
- Assign memory size for our VM (1024 MB sufficient for now).
- Select the option *Use an existing virtual hard disk file* and locate the downloaded VMDK image and create VM
- Now we have to create a NAT Network so go to *File -> Preferences -> Network -> Add a New NAT Network (Click on +)*
- Right click and edit the Network name and CIDR if needed.

- Repeat the process of launching the VM for 2 instances
- Now go to the setting, go to the network setting and change the adapter to NAT Network and select the NAT Network you made
- Launch the VM now
- Install the net-tools to know the IP's of the instance
- create a file and write something into it
- If your file is on the VM with IP **172.168.2.4** and the second VM's IP is **172.168.2.5**.
- Transfer the file using **SCP**

  $ scp tranfer.txt vagrant@172.168.2.5:/home/vagrant
- Check for the file in the Second VM under the **/home/vagrant** directory

Assignment No. 5

- **Title-**
  - Find a procedure to launch virtual machine using try stack (Online Open stack Demo Version)

- **References-**
  - **https://www.amazonaws.cn/en/getting-started/tutorials/launch-a-virtual-machine/**
  - **https://docs.microsoft.com/en-us/azure/virtual-machines/windows/quick-create-portal**
  - **https://cloud.google.com/compute/docs/instances/create-start-instance**

- **Steps-**
  - Launch an Amazon EC2 Instance
  - Configure your Instance
  - Connect to your Instance
  - Terminate Your Instance

Assignment No. 6

- **Title-**
  - Design and deploy a web application in a PaaS environment

- **Steps-**
  - Login to the AWS console
  - Find for AWS Amplify in the services
  - Get Started with Amplify service
  - Click on Host a Web App
  - Then choose to launch it with Github and authenticate your GitHub account for the same
  - After that choose the Repository containing your source code
  - Then Launch the application with the default configurations provided by AWS Amplify

Assignment No. 7

- **Title-**
  - Design and develop custom Application (Mini Project) using Salesforce Cloud
- **Steps-**
  - Start your 30-day free trial of the world's leading CRM

https://www.salesforce.com/eu/form/signup/freetrial-sales-pe/

  - Follow the steps to create application using salesforce.

https://www.youtube.com/watch?app=desktop&v=XL0MSkNSl8E&feature=youtu.be

Assignment No. 8

- **Title-**
  - Design an Assignment to retrieve, verify, and store user credentials using Firebase Authentication, the Google App Engine standard environment, and Google Cloud Data store

- **Steps-**
  - Install required software's as per the requirement
  - Install all the packages which are needed for firebase (firebase-admin , express etc)
  - And follow the steps as per the references-

https://firebase.google.com/docs/reference/admin
https://firebase.google.com/docs/auth/admin
https://firebase.google.com/docs/admin/setup
https://cloud.google.com/appengine/docs/standard/python/configuration-files
https://livebook.manning.com/book/google-cloud-platform-in-action/chapter-11/