Paper Name: Cloud Computing Lab

1. Installation of VMware Workstation on windows OS (version windows 7 to 10)

Here are the steps to install VMware Workstation on a Windows OS (applicable for Windows 7, 8, 8.1, and 10):

Step 1: Download VMware Workstation

1. Open your web browser and go to the VMware website: VMware Workstation.
2. Click on **"Download Now"** for VMware Workstation Pro or **"Try for Free"** if you are downloading the trial version.
3. Save the installer file to your computer.

Step 2: Install VMware Workstation

1. Navigate to the folder where you saved the installer file and double-click on it to launch the installation process.
2. Click **"Next"** on the VMware Workstation Setup Wizard welcome screen.

Step 3: Accept the License Agreement

1. Read the End-User License Agreement (EULA).
2. Click on the radio button to accept the terms and click **"Next"**.

Step 4: Choose Installation Options

1. Select the installation options that suit your needs:
   - **Enhanced Keyboard Driver**: Choose whether you want to install the enhanced keyboard driver.
   - **Check for Product Updates**: Option to check for product updates on startup.
   - **Help improve VMware Workstation**: Option to join the VMware Customer Experience Improvement Program.
2. Click **"Next"**.

Step 5: Select the Destination Folder

1. Choose the installation directory or leave it at the default location.
2. Click **"Next"**.

000611

Step 6: Choose Start Menu Folder

1. Select the Start Menu folder or leave it at the default.
2. Click **"Next"**.

Step 7: Configure Shortcuts

1. Choose if you want to create shortcuts for VMware Workstation on your desktop and/or Start menu.
2. Click **"Next"**.

Step 8: Install VMware Workstation

1. Review your choices and click **"Install"** to begin the installation process.
2. Wait for the installation to complete. This may take a few minutes.

Step 9: Finish Installation

1. Once the installation is complete, click **"Finish"**.
2. You might be prompted to restart your computer. If so, save your work and restart.

Step 10: Enter License Key (Optional)

1. If you have a license key, enter it when prompted. You can also enter the license key later by going to **Help > Enter Serial Number** within VMware Workstation.

Step 11: Launch VMware Workstation

1. After installation, you can launch VMware Workstation from the Start menu or desktop shortcut.
2. Start using VMware Workstation to create and manage virtual machines.

Troubleshooting Tips:

- **Compatibility Issues**: Ensure your Windows OS is up-to-date with the latest service packs and updates.
- **Administrative Privileges**: Run the installer with administrative privileges by right-clicking the installer and selecting "Run as administrator."
- **Software Conflicts**: Disable any antivirus or security software temporarily during installation to prevent conflicts.

With these steps, you should be able to successfully install VMware Workstation on your Windows computer.

2. Installation of C compiler and execute simple programs of C using virtual machine.

To install a C compiler and execute simple C programs using a virtual machine (VM) in VMware Workstation, follow these steps:

Step 1: Set Up the Virtual Machine

1. **Launch VMware Workstation.**
2. **Create a New Virtual Machine:**
   - Select **File > New Virtual Machine.**
   - Choose **Typical (recommended)** and click **Next**.
   - Insert the installation media for the operating system you wish to install (e.g., an ISO file for a Linux distribution like Ubuntu or a Windows ISO) and click **Next**.
   - Follow the prompts to configure and install the operating system.

Step 2: Install a C Compiler

For Linux (e.g., Ubuntu):

1. **Open a Terminal:**
   - You can open the terminal from the Applications menu or by pressing Ctrl + Alt + T.
2. **Update Package List:**

```
bash
Copy code
sudo apt update
```

3. **Install GCC:**

```
bash
Copy code
sudo apt install build-essential
```

4. **Verify Installation:**

```
bash
Copy code
gcc --version
```

For Windows:

1. **Download MinGW:**
   - Go to the MinGW website.
   - Download the mingw-get-setup.exe file.

2. **Install MinGW**:
    - Run the installer.
    - Select the components you want to install (typically, you'll want to select the "mingw32-gcc-g++" package).
    - Follow the installation prompts.
3. **Set Environment Variables**:
    - Open **Control Panel > System and Security > System > Advanced system settings**.
    - Click on **Environment Variables**.
    - Under **System variables**, find the Path variable, select it, and click **Edit**.
    - Add the path to the MinGW bin directory (e.g., C:\MinGW\bin) and click **OK**.
4. **Verify Installation**:
    - Open Command Prompt and type:

```
cmd
Copy code
gcc --version
```

Step 3: Write and Compile a Simple C Program

1. **Create a C Program**:
    - Open a text editor and write a simple C program. For example:

```
c
Copy code
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

2. **Save the Program**:
    - Save the file as hello.c.

For Linux:

1. **Compile the Program**:
    - Open a terminal and navigate to the directory where you saved hello.c.
    - Compile the program with GCC:

```
bash
Copy code
gcc hello.c -o hello
```

2. **Run the Program**:

```
bash
Copy code
./hello
```

For Windows:

1. **Compile the Program**:
   - Open Command Prompt and navigate to the directory where you saved hello.c.
   - Compile the program with GCC:

```
cmd
Copy code
gcc hello.c -o hello.exe
```

2. **Run the Program**:

```
cmd
Copy code
hello.exe
```

Conclusion

By following these steps, you will have set up a virtual machine with a C compiler installed, written a simple C program, and successfully compiled and executed it. This setup can be used for further C programming and development within the VM environment.

3. Create a hello world application and other simple web applications using python/java in Google App Engine

Creating a "Hello World" application and other simple web applications using Python or Java on Google App Engine (GAE) involves a few steps. Here, I'll provide you with the steps and example code for both Python and Java.

Python

Setting Up

1. **Install Google Cloud SDK**: If you haven't already, install the Google Cloud SDK from here.
2. **Initialize Google Cloud SDK**: Run the following command and follow the instructions:

```
sh
Copy code
gcloud init
```

3. **Create a New Project**: If you don't have a project yet, create one:

```
sh
Copy code
```

```
gcloud projects create [YOUR_PROJECT_ID]
gcloud config set project [YOUR_PROJECT_ID]
```

4. **Enable App Engine**: Enable the App Engine service for your project:

```
sh
Copy code
gcloud app create --project=[YOUR_PROJECT_ID]
```

Create a "Hello World" Application

1. **Directory Structure**:

```
css
Copy code
helloworld/
    app.yaml
    main.py
```

2. **app.yaml**: This file tells App Engine how to run your application.

```
yaml
Copy code
runtime: python39  # Use python39 for Python 3.9
entrypoint: python3 main.py

handlers:
- url: /.*
  script: auto
```

3. **main.py**: This file contains your application code.

```
python
Copy code
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return 'Hello, World!'

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```

4. **Deploy the Application**:

```
sh
Copy code
gcloud app deploy
```

5. **View the Application:**

```
sh
Copy code
gcloud app browse
```

More Complex Web Application Example

1. **main.py:**

```
python
Copy code
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/submit', methods=['POST'])
def submit():
    name = request.form['name']
    return f'Hello, {name}!'

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```

2. **index.html** (Place this in a `templates` folder):

```
html
Copy code
<!doctype html>
<html>
    <head>
        <title>Greeting Form</title>
    </head>
    <body>
        <form action="/submit" method="post">
            Name: <input type="text" name="name">
            <input type="submit" value="Submit">
        </form>
    </body>
</html>
```

3. **Deploy the Application:**

```
sh
Copy code
gcloud app deploy
```

Setting Up

1. **Install Google Cloud SDK**: As mentioned earlier.
2. **Initialize Google Cloud SDK**: Run the command:

```
sh
Copy code
gcloud init
```

3. **Create a New Project and Enable App Engine**: As mentioned earlier.

Create a "Hello World" Application

1. **Directory Structure**:

```
css
Copy code
helloworld/
    src/
        main/
            java/
                com/
                    example/
                        appengine/
                            HelloAppEngine.java
            webapp/
                WEB-INF/
                    appengine-web.xml
                    web.xml
    pom.xml
```

2. **pom.xml**:

```
xml
Copy code
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>helloworld</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>war</packaging>

    <dependencies>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.1</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>com.google.cloud.tools</groupId>
                <artifactId>appengine-maven-plugin</artifactId>
                <version>2.2.0</version>
            </plugin>
        </plugins>
    </build>
</project>
```

3. **HelloAppEngine.java**:

```
java
Copy code
package com.example.appengine;

import java.io.IOException;
import javax.servlet.http.*;

public class HelloAppEngine extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws
IOException {
        resp.setContentType("text/plain");
        resp.getWriter().println("Hello, World!");
    }
}
```

4. **appengine-web.xml**:

```
xml
Copy code
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
    <runtime>java11</runtime>
    <instance-class>F1</instance-class>
</appengine-web-app>
```

5. **web.xml**:

```
xml
Copy code
<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>HelloAppEngine</servlet-name>
        <servlet-class>com.example.appengine.HelloAppEngine</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HelloAppEngine</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

6. **Deploy the Application:**

```
sh
Copy code
mvn appengine:deploy
```

7. **View the Application:**

```
sh
Copy code
gcloud app browse
```

More Complex Web Application Example

1. **HelloAppEngine.java:**

```
java
Copy code
package com.example.appengine;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.*;

public class HelloAppEngine extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws
IOException, ServletException {
        req.getRequestDispatcher("/index.jsp").forward(req, resp);
    }

    @Override
    public void doPost(HttpServletRequest req, HttpServletResponse resp) throws
IOException {
        String name = req.getParameter("name");
        resp.setContentType("text/plain");
        resp.getWriter().println("Hello, " + name + "!");
    }
}
```

2. **index.jsp** (Place this in `webapp` folder):

```
jsp
```

000653

```
Copy code
<html>
    <head>
        <title>Greeting Form</title>
    </head>
    <body>
        <form action="/submit" method="post">
            Name: <input type="text" name="name">
            <input type="submit" value="Submit">
        </form>
    </body>
</html>
```

3. **web.xml** (Updated):

```
xml
Copy code
<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>HelloAppEngine</servlet-name>
        <servlet-class>com.example.appengine.HelloAppEngine</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HelloAppEngine</servlet-name>
        <url-pattern>/</url-pattern>
        <url-pattern>/submit</url-pattern>
    </servlet-mapping>
</web-app>
```

4. **Deploy the Application**:

```
sh
Copy code
mvn appengine:deploy
```

These examples should give you a good starting point for deploying simple web applications on Google App Engine using Python and Java. If you need more advanced features or optimizations, you can refer to the official Google Cloud documentation.

4. Launching of web applications using GAE launcher

ogle App Engine Launcher, part of the Google Cloud SDK, was a GUI tool for managing and deploying p Engine applications. However, it has been deprecated, and the recommended method now is to use the mmand-line tools provided by the Google Cloud SDK.

you still have the GAE Launcher and want to use it, here are the general steps for launching web pplications:

Using Google App Engine Launcher

1. **Install Google App Engine SDK**: Download and install the Google App Engine SDK from the Google Cloud SDK download page.
2. **Create a New Application**:
   - Open Google App Engine Launcher.
   - Go to `File` > `Create New Application`.
   - Choose a directory for your application.
   - Set the application ID, which should match the one you created on Google Cloud Console.
   - Set the port number (default is 8080).
3. **Add Application Files**:
   - Add your application files (`app.yaml`, `main.py` or `pom.xml`, `HelloAppEngine.java` etc.) to the directory.
4. **Run the Application Locally**:
   - Select your application in the launcher.
   - Click the `Run` button to start your application locally.
   - Open your web browser and go to `http://localhost:8080` (or the port you specified) to see your application.
5. **Deploy the Application**:
   - Select your application in the launcher.
   - Click the `Deploy` button.
   - Follow the prompts to deploy your application to Google App Engine.

Using Command-Line Tools (Recommended)

1. **Install Google Cloud SDK**: If you haven't installed it already, follow the instructions here.
2. **Initialize the SDK**:

```
sh
Copy code
gcloud init
```

3. **Create a New Project**:

```
sh
Copy code
gcloud projects create [YOUR_PROJECT_ID]
gcloud config set project [YOUR_PROJECT_ID]
```

4. **Enable App Engine**:

000651

```
sh
Copy code
gcloud app create --project=[YOUR_PROJECT_ID]
```

5. **Create Application Files:**
   - Create the directory structure as mentioned in the previous response.
6. **Run the Application Locally:**
   - For Python:

     ```
     sh
     Copy code
     python main.py
     ```

   - For Java:

     ```
     sh
     Copy code
     mvn appengine:run
     ```

   - Open your web browser and go to `http://localhost:8080`.
7. **Deploy the Application:**
   - For Python:

     ```
     sh
     Copy code
     gcloud app deploy
     ```

   - For Java:

     ```
     sh
     Copy code
     mvn appengine:deploy
     ```

   - View your application:

     ```
     sh
     Copy code
     gcloud app browse
     ```

Example with Command-Line Tools

Python

1. **app.yaml:**

```
yaml
Copy code
runtime: python39
entrypoint: python3 main.py
```

```
handlers:
- url: /.*
  script: auto
```

2. **main.py**:

```
python
Copy code
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return 'Hello, World!'

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```

3. **Deploy and Run**:

```
sh
Copy code
gcloud app deploy
gcloud app browse
```

Java

1. **Directory Structure**:

```
css
Copy code
helloworld/
    src/
        main/
            java/
                com/
                    example/
                        appengine/
                            HelloAppEngine.java

            webapp/
                WEB-INF/
                    appengine-web.xml
                    web.xml

    pom.xml
```

2. **pom.xml**:

```
xml
Copy code
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>helloworld</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>war</packaging>

    <dependencies>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.1</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>com.google.cloud.tools</groupId>
                <artifactId>appengine-maven-plugin</artifactId>
                <version>2.2.0</version>
            </plugin>
        </plugins>
    </build>
</project>
```

3. **HelloAppEngine.java**:

```
java
Copy code
package com.example.appengine;

import java.io.IOException;
import javax.servlet.http.*;

public class HelloAppEngine extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws
IOException {
        resp.setContentType("text/plain");
        resp.getWriter().println("Hello, World!");
    }
}
```

4. **appengine-web.xml**:

```
xml
Copy code
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
    <runtime>java11</runtime>
    <instance-class>F1</instance-class>
</appengine-web-app>
```

5. **web.xml**:

```
xml
Copy code
<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>HelloAppEngine</servlet-name>
          <servlet-class>com.example.appengine.HelloAppEngine</servlet-class>
      </servlet>
      <servlet-mapping>
          <servlet-name>HelloAppEngine</servlet-name>
          <url-pattern>/</url-pattern>
      </servlet-mapping>
 </web-app>
```

6. **Deploy and Run**:

```
sh
Copy code
mvn appengine:deploy
gcloud app browse
```

Using the command-line tools provides more flexibility and is more up-to-date with current best practices.

5. Creating a cloud environment using CloudSim and run a scheduling algorithm that is not present in CloudSim

Creating a cloud environment using CloudSim and running a custom scheduling algorithm involves several steps. Below is a step-by-step guide to get you started with CloudSim, including creating a custom scheduling algorithm.

Step 1: Setting Up CloudSim

1. **Download CloudSim**:
   - Visit the CloudSim GitHub repository: CloudSim GitHub

- Download the latest release or clone the repository.

2. **Set Up Your Development Environment**:
   - Make sure you have a Java Development Kit (JDK) installed. You can download the latest JDK from the Oracle website or use OpenJDK.
   - Use an Integrated Development Environment (IDE) like Eclipse or IntelliJ IDEA for easier project management and coding.
3. **Import CloudSim Project into Your IDE**:
   - Open your IDE and create a new Java project.
   - Import the downloaded CloudSim library into your project.

Step 2: Create a Custom Scheduling Algorithm

1. **Understand Existing Schedulers**:
   - Review existing scheduling algorithms in CloudSim to understand how they are implemented. Common classes include `VmAllocationPolicy`, `VmScheduler`, and `CloudletScheduler`.
2. **Implement a Custom Scheduler**:
   - Create a new class that extends one of the base scheduler classes, such as `VmAllocationPolicy`.

Example: Custom VM Allocation Policy

Here's an example of creating a custom VM allocation policy:

```
java
Copy code
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.VmAllocationPolicy;

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

public class CustomVmAllocationPolicy extends VmAllocationPolicy {
    private List<Host> hostList;

    public CustomVmAllocationPolicy(List<? extends Host> list) {
        super(list);
        this.hostList = new ArrayList<>(list);
    }

    @Override
    public boolean allocateHostForVm(Vm vm) {
        for (Host host : hostList) {
            if (host.isSuitableForVm(vm)) {
                return host.vmCreate(vm);
            }
        }
        return false;
    }

    @Override
public void deallocateHostForVm(Vm vm) {
    Host host = vm.getHost();
    if (host != null) {
        host.vmDestroy(vm);
    }
}

@Override
public Host getHost(Vm vm) {
    return vm.getHost();
}

@Override
public Host getHost(int vmId, int userId) {
    for (Host host : hostList) {
        Vm vm = host.getVm(vmId, userId);
        if (vm != null) {
            return host;
        }
    }
    return null;
}
}
```

Step 3: Integrate the Custom Scheduler with CloudSim

1. **Create a Simulation Class**:
   - Create a class that initializes the CloudSim environment, creates datacenters, hosts, VMs, and cloudlets.
2. **Use the Custom Scheduler**:
   - Integrate your custom scheduler into the simulation setup.

Example: Setting Up a CloudSim Simulation

Here's a basic setup for running a CloudSim simulation with the custom scheduler:

```
java
Copy code
import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.core.CloudSim;

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

public class CustomSchedulerExample {
    public static void main(String[] args) {
        try {
            int numUsers = 1;
            Calendar calendar = Calendar.getInstance();
            boolean traceFlag = false;
CloudSim.init(numUsers, calendar, traceFlag);

// Create Datacenter
Datacenter datacenter = createDatacenter("Datacenter_0");

// Create Broker
DatacenterBroker broker = new DatacenterBroker("Broker");

// Create VMs and Cloudlets
List<Vm> vmList = createVMs(broker.getId(), 5);
List<Cloudlet> cloudletList = createCloudlets(broker.getId(), 10);

// Submit VM list to the broker
broker.submitVmList(vmList);

// Submit cloudlet list to the broker
broker.submitCloudletList(cloudletList);

// Start simulation
CloudSim.startSimulation();

List<Cloudlet> newList = broker.getCloudletReceivedList();

CloudSim.stopSimulation();

printCloudletList(newList);
} catch (Exception e) {
    e.printStackTrace();
    System.out.println("Simulation error: " + e.getMessage());
}
}

private static Datacenter createDatacenter(String name) {
    List<Host> hostList = new ArrayList<>();

    int mips = 1000;
    int ram = 2048; // host memory (MB)
    long storage = 1000000; // host storage
    int bw = 10000;

    List<Pe> peList = new ArrayList<>();
    peList.add(new Pe(0, new PeProvisionerSimple(mips)));

    hostList.add(new Host(
        new RamProvisionerSimple(ram),
        new BwProvisionerSimple(bw),
        storage,
        peList,
        new VmSchedulerTimeShared(peList)
    ));

    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

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

        try {
            return new Datacenter(name, characteristics, new
CustomVmAllocationPolicy(hostList), new LinkedList<Storage>(), 0);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    private static List<Vm> createVMs(int userId, int vms) {
        List<Vm> list = new ArrayList<>();

        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

        for (int i = 0; i < vms; i++) {
            list.add(new Vm(i, userId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared()));
        }

        return list;
    }
    private static List<Cloudlet> createCloudlets(int userId, int cloudlets) {
        List<Cloudlet> list = new ArrayList<>();

        long length = 40000;
        int pesNumber = 1;
        long fileSize = 300;
        long outputSize = 300;
        UtilizationModel utilizationModel = new UtilizationModelFull();

        for (int i = 0; i < cloudlets; i++) {
            Cloudlet cloudlet = new Cloudlet(i, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);
            cloudlet.setUserId(userId);
            list.add(cloudlet);
        }

        return list;
    }
```

```
private static void printCloudletList(List<Cloudlet> list) {
    int size = list.size();
    Cloudlet cloudlet;

    String indent = "    ";
    System.out.println();
    System.out.println("========== OUTPUT ==========");
    System.out.println("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);
            System.out.print(indent + cloudlet.getCloudletId() + indent + indent);

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

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

In this example, `CustomVmAllocationPolicy` is a simple custom VM allocation policy, and the `CustomSchedulerExample` class sets up and runs a CloudSim simulation using this custom policy.

You can further customize the VM allocation policy and the simulation setup based on your specific requirements. This example should give you a good starting point for working with CloudSim and implementing custom scheduling algorithms.

6. Transferring the files from one virtual machine to another virtual machine

Transferring files between virtual machines (VMs) can be achieved using several methods depending on the operating systems involved, network setup, and specific tools available. Here, I will outline a few common methods for both Linux and Windows VMs.

Method 1: Using SCP (Secure Copy Protocol)

Linux to Linux

1. **Install SSH**: Ensure that SSH is installed and running on both VMs.

```
sh
Copy code
sudo apt-get update
sudo apt-get install openssh-server
sudo systemctl start ssh
sudo systemctl enable ssh
```

2. **Transfer Files**:

```
sh
Copy code
scp /path/to/local/file username@remote_host:/path/to/remote/destination
```

Example:

```
sh
Copy code
scp /home/user/file.txt user@192.168.1.2:/home/user/
```

Linux to Windows

1. **Install SSH Server on Windows**: If not already installed, install an SSH server on the Windows machine. OpenSSH can be installed via PowerShell:

```
powershell
Copy code
Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0
Start-Service sshd
Set-Service -Name sshd -StartupType 'Automatic'
```

2. **Transfer Files**:

```
sh
Copy code
scp /path/to/local/file username@windows_host:/path/to/remote/destination
```

Example:

```
sh
Copy code
scp /home/user/file.txt user@192.168.1.3:C:/Users/user/
```

Method 2: Using rsync (for Linux VMs)

1. **Install rsync**: Ensure that `rsync` is installed on both VMs.

```
sh
Copy code
sudo apt-get install rsync
```

2. **Transfer Files:**

```
sh
Copy code
rsync -avz /path/to/local/file username@remote_host:/path/to/remote/destination
```

Example:

```
sh
Copy code
rsync -avz /home/user/file.txt user@192.168.1.2:/home/user/
```

Method 3: Using SFTP (Secure File Transfer Protocol)

1. **Install SFTP Client**: Use an SFTP client like `sftp` command-line tool or graphical clients like FileZilla.
2. **Transfer Files:**

```
sh
Copy code
sftp username@remote_host
put /path/to/local/file /path/to/remote/destination
```

Example:

```
sh
Copy code
sftp user@192.168.1.2
put /home/user/file.txt /home/user/
```

Method 4: Using Cloud Storage

If your VMs are hosted on a cloud provider (like AWS, GCP, or Azure), you can use cloud storage as an intermediary:

AWS S3

1. **Upload to S3:**

```
sh
Copy code
aws s3 cp /path/to/local/file s3://your-bucket-name/path/to/destination
```

2. **Download from S3**:

```
sh
Copy code
aws s3 cp s3://your-bucket-name/path/to/destination /path/to/local/destination
```

GCP Storage

1. **Upload to GCP Storage**:

```
sh
Copy code
gsutil cp /path/to/local/file gs://your-bucket-name/path/to/destination
```

2. **Download from GCP Storage**:

```
sh
Copy code
gsutil cp gs://your-bucket-name/path/to/destination /path/to/local/destination
```

Method 5: Using Shared Network Folders

Windows to Windows

1. **Set Up Shared Folder**: Share a folder on the source VM and set appropriate permissions.
2. **Access Shared Folder**: On the destination VM, map the shared folder or directly access it using the network path:

```
cmd
Copy code
net use X: \\remote_host\shared_folder
copy X:\file.txt C:\destination
```

Linux to Linux

1. **Install NFS**:

```
sh
Copy code
sudo apt-get install nfs-kernel-server
```

2. **Export Directory**: Edit `/etc/exports` on the source VM:

```
bash
Copy code
/path/to/share *(rw,sync,no_root_squash,no_subtree_check)
```

3. **Mount Shared Directory**:

```
sh
Copy code
sudo mount remote_host:/path/to/share /path/to/mount
```

Example: Transferring Files with SCP

Scenario: Transferring `example.txt` *from VM1 to VM2*

1. **On VM1 (Source)**:

   ```
   sh
   Copy code
   scp /home/user/example.txt user@192.168.1.2:/home/user/
   ```

2. **On VM2 (Destination)**: Verify the file transfer:

   ```
   sh
   Copy code
   ls /home/user/
   ```

Conclusion

These methods provide a variety of ways to transfer files between VMs depending on your specific setup and requirements. If you are using a cloud provider, leveraging their storage solutions can simplify the process. For on-premise VMs or those without cloud storage, SCP, rsync, and SFTP are reliable options.

7. Find a procedure to launch virtual machine using trystack (Online Openstack Demo Version)

TryStack is an online OpenStack demo environment that allows users to test and explore OpenStack functionalities without the need to install anything locally. Here's a step-by-step guide to launching a virtual machine (VM) using TryStack:

Step 1: Sign Up and Log In

1. **Sign Up for TryStack**:
   - Go to the TryStack website: TryStack.
   - Sign up for an account using your email or social media accounts.

2. **Log In to TryStack:**
   - Log in to your TryStack account with your credentials.

Step 2: Access the Horizon Dashboard

1. **Navigate to the Horizon Dashboard:**
   - After logging in, you will be redirected to the OpenStack Horizon Dashboard. This is the web-based interface for managing your OpenStack resources.

Step 3: Launch an Instance

1. **Navigate to the Instances Page:**
   - From the left-hand menu, click on "Project" > "Compute" > "Instances".
2. **Launch an Instance:**
   - Click the "Launch Instance" button at the top of the page.
3. **Configure Your Instance:**
   - **Details:**
     - **Instance Name:** Enter a name for your instance.
     - **Availability Zone:** Select an availability zone (if applicable).
   - **Source:**
     - **Select Boot Source:** Choose "Image".
     - **Create New Volume:** Select "No".
     - **Image Name:** Select an image from the available list (e.g., "Ubuntu", "CentOS").
   - **Flavor:**
     - **Select Flavor:** Choose a flavor that matches your requirements (e.g., `m1.tiny` for a small VM).
   - **Networks:**
     - **Network:** Select the network to which your instance will connect. This is typically preconfigured in TryStack.
   - **Security Groups:**
     - **Security Group:** Select a security group that has appropriate rules for accessing your instance (e.g., allow SSH and HTTP).
   - **Key Pair:**
     - **Key Pair:** Select an existing key pair or create a new one. This is used for SSH access to your instance. If creating a new key pair, make sure to download the private key file (`.pem`).
4. **Launch:**
   - Click the "Launch" button at the bottom right corner.

Step 4: Access Your Instance

1. **Check Instance Status:**
   - Wait for your instance to move from the "Spawning" state to the "Active" state. This may take a few minutes.
2. **Obtain Instance Details:**