1. **Aim:**
   To install VirtualBox/VMware Workstation and set up a Linux or Windows OS on a Windows 7/8 system.

Procedure:

1. Visit https://www.virtualbox.org.
2. Click **Download VirtualBox** → Select **Windows hosts**.
3. Once downloaded, open the EXE file.
4. Follow the installation steps:
    - Click **Next** on initial prompts
    - Click **Yes** when prompted
    - Click **Install**, then **Finish**

Result:
Oracle VM VirtualBox is successfully installed on Windows.

2. Aim:
To create a virtual machine using VirtualBox/VMware Workstation on Windows 7 or 8.

Procedure:

1. Open VirtualBox and click **New**.
2. Name the VM (e.g., *ubuntu1404*); type and version auto-fill as **Linux** and **Ubuntu (64-bit)**.
3. Allocate memory (e.g., 8192 MB based on your system RAM).
4. Select **Create a virtual hard drive now** → Click **Create**.
5. Choose **VDI** as the disk file type → Click **Next**.
6. Select **Fixed size** for better performance.

7. Set disk size (e.g., 100 GB) → Click **Create**.
8. To install Ubuntu, go to **Settings** → **Storage**, click **Empty** under Controller: IDE, then click the CD icon and choose your Ubuntu ISO file.
9. Start the VM → Click **Install Ubuntu**.
10. Select **Erase disk and install Ubuntu** → Continue with the defaults.
11. Enter your name, username, and password → Click **Continue**.
12. After installation, click **Restart Now**, remove the installation media, and press **Enter**.
13. Log in using your password.

Result:
The virtual machine with Ubuntu is successfully created and installed on Windows.

3. Aim:
To install a C compiler in a virtual machine and execute a C program.

Procedure:

- Open the terminal in the guest operating system (e.g., Ubuntu).
- Install the C compiler by running:
    - `sudo apt install gcc`
    - `sudo apt install build-essential`
- Navigate to the Desktop:
    - `cd Desktop`
- Create a new C file:
    - `touch hello.c`

- Open the file in a text editor and write a sample C program. Save the file.
- Compile the program:
    - `gcc hello.c`
- Run the compiled program:
    - `./a.out`

Result:
C compiler installed and a sample C program executed successfully in the virtual machine.

4. AIM:
To use Google App Engine (GAE) launcher to launch web applications.

PROCEDURE:

- **STEP 1:** Create a new Cloud Console project or retrieve the Project ID of an existing project to use.
- **STEP 2:** Go to the project page on Google Cloud Console.
- **STEP 3:** Install and initialize the Google Cloud SDK.
- **STEP 4:** Download the Google Cloud SDK.
- **STEP 5:** Create a website to host on Google App Engine.
- **STEP 6:** Understand the basic structure for the project.
- **STEP 7:** The project uses the following structure:
    - `app.yaml` — Configures the settings of your App Engine application.

    - `www/` — Directory to store all static files such as HTML, CSS, images, and JavaScript.
    - `css/` — Directory to store stylesheets.
    - `style.css` — Basic stylesheet to format your website's look and feel.
    - `images/` — Optional directory to store images.
    - `index.html` — HTML file that displays content for your website.
    - `js/` — Optional directory to store JavaScript files.
    - Other asset directories as needed.
- **STEP 8:** Creating the `app.yaml` file:
    - The `app.yaml` file tells App Engine how to map URLs to your static files.
    - Create a directory with the same name as your project ID (found in Google Cloud Console).
    - Inside that directory, create a file named `app.yaml`.

Edit the `app.yaml` file to include the following:

```
runtime: python27
api_version: 1
threadsafe: true

handlers:
- url: /
  static_files: www/index.html
  upload: www/index.html
- url: /(.*)
  static_files: www/\1
  upload: www/(.*)
```

    -

- **STEP 9:** Creating the `index.html` file:
    - Create an HTML file inside the `www` directory.

Use the following sample HTML content:

```
<html>
<head>
  <title>Hello, world!</title>
  <link rel="stylesheet" type="text/css" href="/css/style.css">
</head>
<body>
  <h1>Hello, world!</h1>
  <p>This is a simple static HTML file that will be served from Google
App Engine.</p>
</body>
</html>
```

    -
- **STEP 10:** Deploying your application to App Engine:

From the root directory where your `app.yaml` file is located, run:

```
gcloud app deploy
```

    -
    - Optional flags:
        - Use `--project [YOUR_PROJECT_ID]` to specify a different project ID.
        - Use `-v [YOUR_VERSION_ID]` to specify a version ID.
- **STEP 11:** Viewing your application:

Launch your browser and open your app with the command:

```
gcloud app browse
```

    -

- This opens the app at the URL:
  `https://PROJECT_ID.REGION_ID.r.appspot.com`

RESULT:
Thus, the Google App Engine launcher is successfully used to launch web applications.

5 and 6. AIM:

To simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.

PROCEDURE:

- **Step 1:** Download Eclipse IDE for Windows 64-bit from

  https://www.eclipse.org/downloads/packages/release/kepler/sr1/eclipse-ide-java-developers
  and install it on your local machine.

- **Step 2:** Download CloudSim version 3.0.3 from GitHub repository:
  https://github.com/Cloudslab/cloudsim/releases/tag/cloudsim-3.0.3

- **Step 3:** Download Apache Commons Math 3.6.1 library from:
  https://commons.apache.org/proper/commons-math/download_math.cgi

- **Step 4:** Extract the downloaded CloudSim 3.0.3 and Apache Commons Math 3.6.1 ZIP files to a local directory.

- **Step 5:** Navigate to the extracted Eclipse folder and launch `eclipse.exe`.

- **Step 6:** In Eclipse, go to `File` -> `New` -> `Project` to open the New Project wizard.

- **Step 7:** Select `Java Project` from the options and click `Next`.

- **Step 8:** In the new project window, set the **Project Name** to `CloudSim`.

- **Step 9:** Uncheck `Use default location`, then click `Browse` and select the folder where you extracted CloudSim source code.
- **Step 10:** Ensure you can see folders like `bin`, `docs`, `examples` in the selected directory and then click `Next`.
- **Step 11:** Proceed to the project settings window.
- **Step 12:** Open the `Libraries` tab. If `commons-math3-3.x.jar` is not present, click `Add External JARs`.
- **Step 13:** Browse to the location where you extracted Apache Commons Math 3.6.1, select the `commons-math3-3.x.jar` file, and click `Open`.
- **Step 14:** Confirm that the jar is listed and click `Finish`. Eclipse will start configuring the project (may take 2–3 minutes).
- **Step 15:** Once configured, open `Project Explorer` in Eclipse and start exploring the CloudSim project. Eclipse may build the workspace automatically on first load.
- **Step 16:** Navigate to `examples` folder inside the project explorer, expand `org.cloudbus.cloudsim.examples` package, and open `CloudSimExample1.java`.
- **Step 17:** Run the example using `Run` -> `Run` menu or by pressing `Ctrl + F11`.
- **Step 18:** On successful execution, observe the output in Eclipse console window displaying the simulation results.

RESULT:

CloudSim is successfully simulated using the Eclipse IDE environment, allowing you to run and modify cloud simulation scenarios, including custom scheduling algorithms not present by default.

Procedure to Import Eclipse and Run Scheduling Algorithms in Your System

Step 1:
Download Eclipse IDE for Windows 64-bit from:
https://www.eclipse.org/downloads/packages/release/kepler/sr1/eclipse-ide-java-developers
and install it on your local machine.

Step 2:
Download the scheduling source code `cloudsim-code-master` from the GitHub repository:
https://github.com/shiro873/Cloudsim-Code
and save it to your local machine.

Step 3:
Download Apache Commons Math 3.6.1 from:
https://commons.apache.org/proper/commons-math/download_math.cgi
and save it locally.

Step 4:
Extract the downloaded CloudSim source code (`cloudsim-code-master`) and Apache Commons Math 3.6.1 libraries.

Step 5:
Navigate to the folder where you extracted Eclipse and launch `eclipse.exe`.

Step 6:
In Eclipse, go to the menu: `File` -> `New` -> `Project` to open the New Project wizard.

Step 7:
In the wizard, select `Java Project` and click `Next`.

Step 8:
In the New Project window, set the Project Name as:
`CloudSim`

Step 9:
Uncheck `Use default location`. Click `Browse` and select the folder where you extracted the `cloudsim-code-master` source code. Click `Next`.

Step 10:
Make sure the selected folder contains the typical CloudSim folders like `bin`, `docs`, `examples`, etc.

Step 11:
Click `Next` to continue to the project settings.

Step 12:
Once the project is configured, open the `Project Explorer`. Eclipse will automatically build the workspace, which may take some time depending on your system.

The final screen you will see confirms that CloudSim is successfully imported.

Step 13:
To test, navigate to the `src` folder, expand the default package, and open `RoundRobin.java`.

Step 14:
Run the file by selecting `Run` -> `Run` from the menu or pressing `Ctrl + F11`.
If successful, output related to the scheduling algorithm execution will display in the Eclipse console.

Result:

The scheduling algorithm was executed successfully, and CloudSim was simulated using the Eclipse environment.

7. AIM:

To transfer files between virtual machines.

PROCEDURE:

Three ways to transfer files:

1. **Copy and Paste:**

- In VirtualBox, go to **Devices > Drag and Drop** and select **Bidirectional** to enable copying files between host and guest.

2. **USB Drive:**

- Install the **VirtualBox Extension Pack** from virtualbox.org.

- Enable USB support in **Settings > USB**.
- Plug in a USB device to transfer files via the VM.

3. **Shared Folder:**

- Install **Guest Additions** via **Devices > Insert Guest Additions CD image** in the VM.
- Go to **Devices > Shared Folders > Shared Folder Settings**, add a folder from the host, and enable **Auto-mount** and **Make Permanent**.
- Access shared folder inside the VM to transfer files.

RESULT:

Files can be successfully transferred between virtual machines using these methods.

8. **AIM:**

To install OpenStack on a CentOS 7 virtual machine using Oracle VirtualBox and PackStack.

PROCEDURE:

Step 1: Download and install Oracle VirtualBox and the Extension Pack from
https://virtualbox.org/wiki/downloads

Step 2: Download CentOS 7 OVA (Open Virtual Appliance) image from
https://linuxvmimages.com/images/centos-7
Import the OVA into Oracle VirtualBox.

Step 3: Start the CentOS 7 VM and login using:

- Username: `centos`
- Password: `centos`
  Switch to root user in terminal:

```
sudo su -
```

Step 4: Disable and stop the firewall service:

```
systemctl disable firewalld
systemctl stop firewalld
```

Step 5: Disable and stop NetworkManager service:

```
systemctl disable NetworkManager
systemctl stop NetworkManager
```

Step 6: Enable and start the network service:

```
systemctl enable network
systemctl start network
```

Step 7: Enable RDO repository for OpenStack:

```
yum install -y https://rdoproject.org/repos/rdo-release.rpm
```

Step 8: Update current packages:

```
yum update -y
```

Step 9: Install OpenStack PackStack:

```
yum install -y openstack-packstack
```

Step 10: Deploy OpenStack all-in-one setup:

```
packstack --allinone
```

Step 11: View OpenStack admin credentials:

```
cat keystonerc_admin
```

Step 12: Open the OpenStack dashboard URL shown in the output and login with the credentials from the previous step.

RESULT:

OpenStack is successfully installed and launched on the CentOS 7 virtual machine.