Submitted in partial fulfillment of the
Requirements for the award of the Degree of

MASTER OF SCIENCE (INFORMATION TECHNOLOGY)

By

POJAARY MANISH SUDHAKAR
SeatNumber:(4133186)

SUBJECT NAME : MICROSERVICES ARCHITECTURE AND IMAGE PROCESSING

DEPARTMENT OF INFORMATION TECHNOLOGY

N. G. ACHARYA & D. K. MARATHE COLLEGE

(Affiliated to University of Mumbai)

MUMBAI , 400071

MAHARASHTRA

2022-23

DEPARTMENT OF INFORMATION TECHNOLOGY

N. G. ACHARYA & D. K. MARATHE COLLEGE

(Affiliated to University of Mumbai)

MUMBAI – MAHARASHTRA - 400071

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that Poojary Manish Sudhakar bearing SeatNo:4133186 submitted journal of data science and research in computing in partial fulfillment of the requirements for the award of Degree of MASTER OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Internal Guide **Coordinator**

External Examiner

Date: **College Seal**

INDEX

Sr. No.	Date	Title	Signature
1.	19/03/2022	Create a console based ASP.net core application	
2.	24/03/2022	Create a MVC Project in ASP.net core	
3.	31/03/2022	Usage of Docker Desktop	
4.	07/04/2022	Working with Docker	
5.	16/04/2022	Building ASP.Net core REST API	
6.	21/04/2022	Working with Circle CI for continuous integration	
7.	22/04/2022	Working with Team Service	

Practical No 1

Aim : Create a console based ASP.net core application.

Source Code :

Step 1 :

- Download the asp.net core sdk from https://dotnet.microsoft.com/learn/dotnet/hello-worldtutorial/install
- Install the asp.net core sdk.
- To check whether the asp.net sdk is successful install, open command prompt and type command: **dotnet**

```
C:\Users\Shraddha Shah>dotnet

Usage: dotnet [options]
Usage: dotnet [path-to-application]

Options:
  -h|--help         Display help.
  --info            Display .NET information.
  --list-sdks       Display the installed SDKs.
  --list-runtimes   Display the installed runtimes.

path-to-application:
  The path to an application .dll file to execute.

C:\Users\Shraddha Shah>
```

- To check the version of the dotnet

```
C:\Users\Shraddha Shah>dotnet --version
6.0.202
```

Step 2 :

- Go to the drive where you want to create the console application. Create a folder in the drive and go to that folder. Type the following command in the command prompt to create the application.

```
D:\MSA Pracs\prac1>cd..

D:\MSA Pracs>md HelloWorld

D:\MSA Pracs>cd Hell*

D:\MSA Pracs\HelloWorld>dotnet new console
The template "Console App" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on D:\MSA Pracs\HelloWorld\HelloWorld.csproj...
  Determining projects to restore...
  Restored D:\MSA Pracs\HelloWorld\HelloWorld.csproj (in 63 ms).
Restore succeeded.
```

- Restore the project and run the application

```
D:\MSA Pracs>cd hell*

D:\MSA Pracs\HelloWorld>dotnet restore
  Determining projects to restore...
  Restored D:\MSA Pracs\HelloWorld\HelloWorld.csproj (in 25.91 sec).

D:\MSA Pracs\HelloWorld>dotnet run
Hello, World!
```

Step 3 :

- Now open **HelloWorld.csproj** file, edit the code

```
<Project Sdk="Microsoft.NET.Sdk">
        <PropertyGroup>
        <OutputType>Exe</OutputType>
        <TargetFramework>net6.0</TargetFramework>
        <ImplicitUsings>enable</ImplicitUsings>
        <Nullable>enable</Nullable>
        </PropertyGroup>
        <ItemGroup>
        <PackageReference Include="Microsoft.AspNetCore.Mvc"
Version="1.1.1"/>
        <PackageReference Include="Microsoft.AspNetCore.Server.Kestrel"
Version="1.1.1"/>
        <PackageReference Include="Microsoft.Extensions.Logging"
Version="1.1.1"/>
        <PackageReference Include="Microsoft.Extensions.Logging.Console"
Version="1.1.1"/>
        <PackageReference Include="Microsoft.Extensions.Logging.Debug"
Version="1.1.1"/>
        <PackageReference
Include="Microsoft.Extensions.Configuration.CommandLine"
Version="1.1.1"/>
        </ItemGroup>
</Project>
```

- Open Program.cs file and edit the code

```
        using System;
        using Microsoft.AspNetCore.Builder;
        using Microsoft.AspNetCore.Hosting;
        using Microsoft.Extensions.Logging;
        using Microsoft.AspNetCore.Http;
        using Microsoft.Extensions.Configuration;
        namespace HelloWorld // Note: actual namespace depends on the
project name.
        {
        internal class Program
            {
```

```
                    static void Main(string[] args)
{
        var config = new ConfigurationBuilder()
           .AddCommandLine(args)
   .Build();
        var host = new WebHostBuilder()
         .UseKestrel()
         .UseStartup<Startup>()
         .UseConfiguration(config)
 .Build();
        host.Run();
        }
            }
        public class Startup
        {
        public Startup(IHostingEnvironment env) { }
        public void Configure(IApplicationBuilder app, IHostingEnvironment
env, ILoggerFactory loggerFactory)
        {
        app.Run(async (context) => { await
context.Response.WriteAsync("Hello, world!");});
        }
        }
        }
```

Step 4 :

Restore the project.

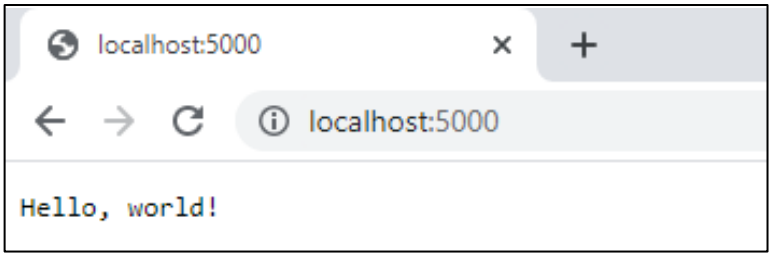
```
D:\MSA Pracs\HelloWorld>dotnet restore
  Determining projects to restore...
  All projects are up-to-date for restore.
```

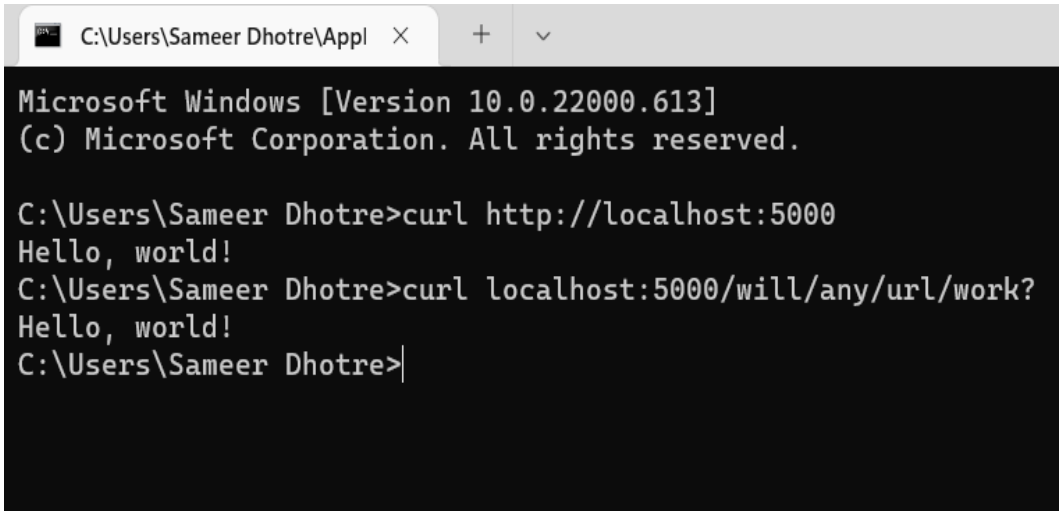
Output :

Run the application

```
D:\MSA Pracs\HelloWorld>dotnet run
Hosting environment: Production
Content root path: D:\MSA Pracs\HelloWorld\bin\Debug\net6.0\
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
Application is shutting down...
```

Now open the browser open the url: http://localhost:5000


localhost:5000
localhost:5000
Hello, world!


C:\Users\Sameer Dhotre\Appl
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Sameer Dhotre>curl http://localhost:5000
Hello, world!
C:\Users\Sameer Dhotre>curl localhost:5000/will/any/url/work?
Hello, world!
C:\Users\Sameer Dhotre>

Practical No 2

Aim : Create a MVC Project in ASP.net core

Source Code :

Step 1 :

Create a mvc project

dotnet new mvc –auth none

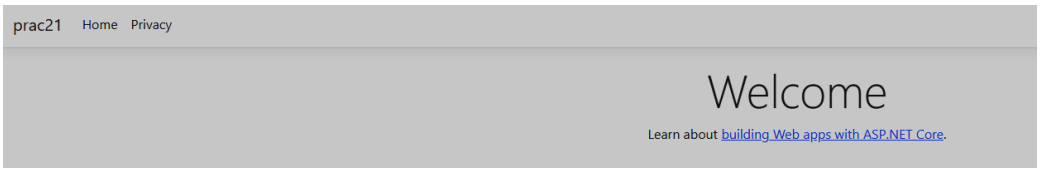
```
D:\Microservices Architecture\Practice Practical\Practs\pracs2>dotnet new mvc --auth none
The template "ASP.NET Core Web App (Model-View-Controller)" was created successfully.
This template contains technologies from parties other than Microsoft, see https://aka.ms/aspnetcore/6.0-third-party-not
ices for details.

Processing post-creation actions...
Running 'dotnet restore' on D:\Microservices Architecture\Practice Practical\Practs\pracs2\pracs2.csproj...
  Determining projects to restore...
  Restored D:\Microservices Architecture\Practice Practical\Practs\pracs2\pracs2.csproj (in 278 ms).
Restore succeeded.


D:\Microservices Architecture\Practice Practical\Practs\pracs2>
```

Step 2 :

Restore, build and run the program.

Use the first url of the command prompt in the browser and see theoutput

```
D:\Microservices Architecture\Practice Practical\Practs\pracs2>dotnet build
Microsoft (R) Build Engine version 17.1.1+a02f73656 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

  Determining projects to restore...
  All projects are up-to-date for restore.
  pracs2 -> D:\Microservices Architecture\Practice Practical\Practs\pracs2\bin\Debug\net6.0\pracs2.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:04.72

D:\Microservices Architecture\Practice Practical\Practs\pracs2>dotnet run
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7091
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5103
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: D:\Microservices Architecture\Practice Practical\Practs\pracs2\
```



Step 3 :

Go to Models Folder and create StockQuote.cs file in it.

```
using System;
namespace pracs.Models
{
    public class StockQuote
```

```
    {
            public string Symbol {get;set;}
            public int Price{get;set;}
    }
}
```

Step 4 :

Now go to views folder and then in home folder. Edit the index.cshtml file

```
@{
   ViewData["Title"] = "Home Page";
}

<div class="text-center">
   <h1 class="display-4">Welcome</h1>
   Symbol: @Model.Symbol <br/>
    Price: $@Model.Price <br/>
</div>
```

Step 5 :

Now go to controller folder and edit HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using pracs2.Models;

namespace pracs2.Controllers;

public class HomeController : Controller
{
   public async Task <IActionResult> Index()
    {
            var model= new StockQuote{ Symbol="Nike", Price=3200};
            return View(model);
    }
```

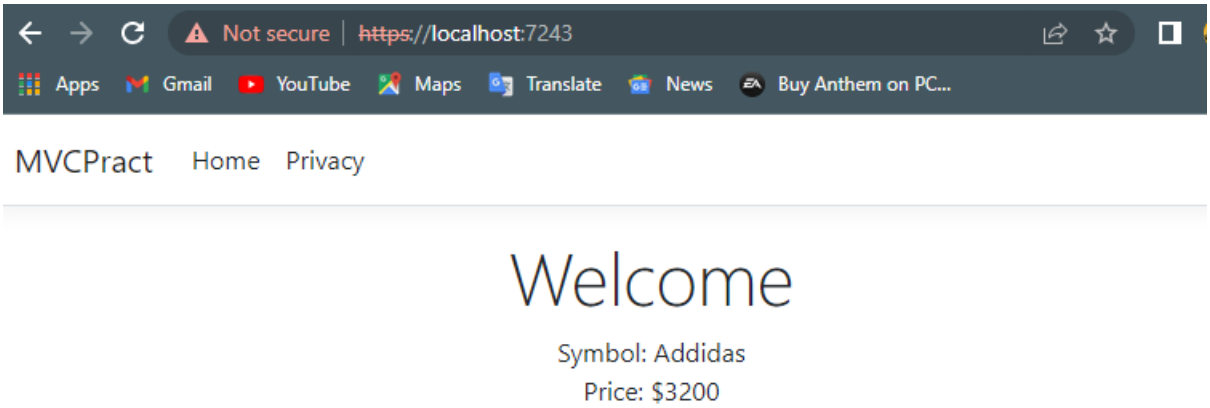
}

Step 6 :

```
D:\Microservices Architecture\Practice Practical\Practs\pracs2>dotnet build
Microsoft (R) Build Engine version 17.1.1+a02f73656 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

  Determining projects to restore...
  All projects are up-to-date for restore.
  pracs2 -> D:\Microservices Architecture\Practice Practical\Practs\pracs2\bin\Debug\net6.0\pracs2.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:04.31

D:\Microservices Architecture\Practice Practical\Practs\pracs2>dotnet run
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7091
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5103
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: D:\Microservices Architecture\Practice Practical\Practs\pracs2\
```

Output :

Open the first url in the browser and see the output



Practical No 3

Aim : Usage of Docker Desktop

Commands & its output :

Open command prompt

- To check whether docker is installed properly
  $ docker

```
D:\msa>docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default
                           "C:\\Users\\Admin\\.docker")
  -c, --context string     Name of the context to use to connect to the
                           daemon (overrides DOCKER_HOST env var and
                           default context set with "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level
                           ("debug"|"info"|"warn"|"error"|"fatal")
                           (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default
                           "C:\\Users\\Admin\\.docker\\ca.pem")
      --tlscert string     Path to TLS certificate file (default
                           "C:\\Users\\Admin\\.docker\\cert.pem")
      --tlskey string      Path to TLS key file (default
                           "C:\\Users\\Admin\\.docker\\key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx (Docker Inc., v0.8.2)
  compose*    Docker Compose (Docker Inc., v2.4.1)
  config      Manage Docker configs
  container   Manage containers
  context     Manage contexts
  image       Manage images
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  sbom*       View the packaged-based Software Bill Of Materials (SBOM) for an image (Anchore Inc., 0.6.0)
  scan*       Docker Scan (Docker Inc., v0.17.0)
  secret      Manage Docker secrets
  service     Manage services
  stack       Manage Docker stacks
  swarm       Manage Swarm
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes
```

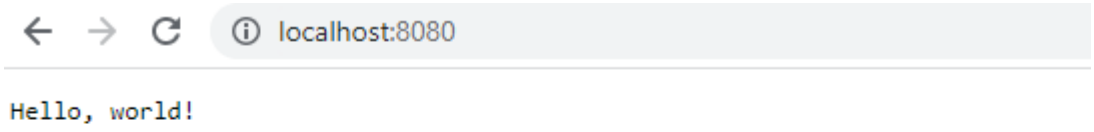
- To see the version of the docker
  $ docker -v

```
D:\msa>docker -v
Docker version 20.10.14, build a224086

D:\msa>
```

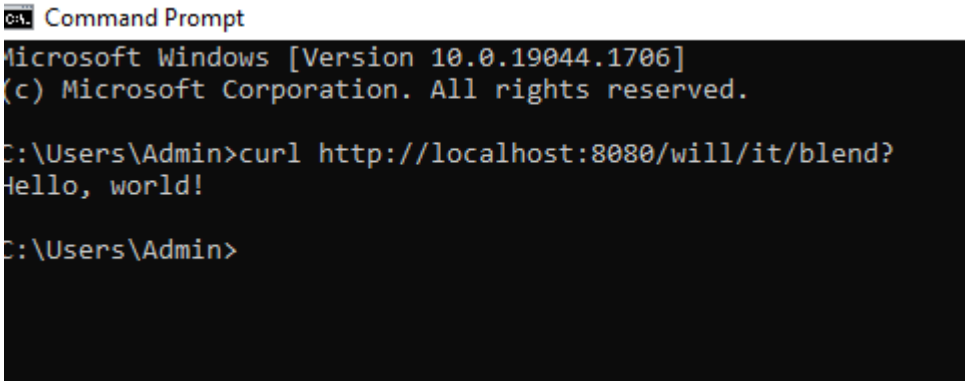
- To run hello-world image
  $ docker run -p 8080:8080 dotnetcoreservices/hello-world

```
D:\msa>docker run -p 8080:8080 dotnetcoreservices/hello-world
Hosting environment: Production
Content root path: /pipeline/source/app/publish
Now listening on: http://0.0.0.0:8080
Application started. Press Ctrl+C to shut down.
```

- Run localhost in the browser
  http://localhost:8080



- To see the output in the command prompt
  $ curl http://localhost:8080/will/it/blend?



- To see the images in the docker
  $ docker ps

```
C:\Users\Admin>docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED          STATUS          PORTS                    NAMES
35c840e18b74   dotnetcoreservices/hello-world       "/pipeline/source/ap…"   3 minutes ago    Up 3 minutes    0.0.0.0:8080->8080/tcp   thirsty_mendeleev
19e44394ce14   8c2c38aa676e                         "/kube-vpnkit-forwar…"   6 minutes ago    Up 6 minutes                             k8s_vpnkit-controller_vpnkit-controller_kube-system_3b9c1da2-6cd8-4f58-9994-292c60ec1c30_40
cf2b29cf594c   99f80471f470                         "/storage-provisione…"   7 hours ago      Up 7 hours                               k8s_storage-provisioner_storage-provisioner_kube-system_026bff3b-ff08-427e-9070-335e41f0a842_4
12d554ff9013   8d147537fb7d                         "/coredns -conf /etc…"   7 hours ago      Up 7 hours                               k8s_coredns_coredns-78fcd69978-sc49b_kube-system_5bf26f98-4b52-4e5b-8dbf-ee5adca62771_2
430216e1dc87   8d147537fb7d                         "/coredns -conf /etc…"   7 hours ago      Up 7 hours                               k8s_coredns_coredns-78fcd69978-xhwnz_kube-system_6f19c380-8277-4bae-a972-7a1856305d19_2
2328fe28f628   8f8fdd6672d4                         "/usr/local/bin/kube…"   7 hours ago      Up 7 hours                               k8s_kube-proxy_kube-proxy-2494n_kube-system_b1ba30bc-f010-47d6-9e15-c92d1cfd1d21_2
185f06a670e7   k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago      Up 7 hours                               k8s_POD_storage-provisioner_kube-system_026bff3b-ff08-427e-9070-335e41f0a842_2
094ffbf4d103   k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago      Up 7 hours                               k8s_POD_coredns-78fcd69978-xhwnz_kube-system_6f19c380-8277-4bae-a972-7a1856305d19_2
f38718d95ce1   k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago      Up 7 hours                               k8s_POD_vpnkit-controller_kube-system_3b9c1da2-6cd8-4f58-9994-292c60ec1c30_2
7c3be6de03ca   k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago      Up 7 hours                               k8s_POD_coredns-78fcd69978-sc49b_kube-system_5bf26f98-4b52-4e5b-8dbf-ee5adca62771_2
64da36c8a371   k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago      Up 7 hours                               k8s_POD_kube-proxy-2494n_kube-system_b1ba30bc-f010-47d6-9e15-c92d1cfd1d21_2
81b781a4a6bd   004811815584                         "etcd --advertise-cl…"   7 hours ago      Up 7 hours                               k8s_etcd_etcd-docker-desktop_kube-system_e5422d6ad44165160915 3d62d1ad42f6_2
fa53e5984047   059e6cd8cf78                         "kube-apiserver --ad…"   7 hours ago      Up 7 hours                               k8s_kube-apiserver_kube-apiserver-docker-desktop_kube-system_3ca039f7ff0c547d271c37587418b712_2
08a2fd1e5e54   04185bc88e08                         "kube-controller-man…"   7 hours ago      Up 7 hours                               k8s_kube-controller-manager_kube-controller-manager-docker-desktop_kube-system_1fe1f00ef31eb1efe8731cc2f5b77
6aa_3
9efb2029ccfea   935d8fdc2d52                         "kube-scheduler --au…"   7 hours ago      Up 7 hours                               k8s_kube-scheduler_kube-scheduler-docker-desktop_kube-system_d19d89767cc69ae3010bc658c1cd2e46_2
8efb5414ae38   k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago      Up 7 hours                               k8s_POD_etcd-docker-desktop_kube-system_e5422d6ad44165160915 3d62d1ad42f6_2
66ddfbac9d10   k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago      Up 7 hours                               k8s_POD_kube-apiserver-docker-desktop_kube-system_3ca039f7ff0c547d271c37587418b712_2
c1d121560a2    k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago      Up 7 hours                               k8s_POD_kube-controller-manager-docker-desktop_kube-system_1fe1f00ef31eb1efe8731cc2f5b776aa_2
5ce6bf1169eb   k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago      Up 7 hours                               k8s_POD_kube-scheduler-docker-desktop_kube-system_d19d89767cc69ae3010bc658c1cd2e46_2
```

- To terminate the image in the docker.
  note the container id of the docker that you want to terminal and replace the <Containerid> in the below command

$ docker kill <containerid>

```
C:\Users\Admin>docker kill 35c840e18b74
35c840e18b74
```

To check whether the docker is terminated or not

$ docker ps

```
C:\Users\Admin>docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED              STATUS              PORTS     NAMES
675693bab0bb   8c2c38aa676e                         "/kube-vpnkit-forwar…"   About a minute ago   Up About a minute             k8s_vpnkit-controller_vpnkit-controller_kube-system_3b9c1da2-6cd8-4f58-9994-292c60ec1c30_41
cf2b29cf594c   99f80471f470                         "/storage-provisione…"   7 hours ago          Up 7 hours                    k8s_storage-provisioner_storage-provisioner_kube-system_026bff3b-ff00-427e-9070-335e41f0a842_4
12d554ff9013   8d147537fb7d                         "/coredns -conf /etc…"   7 hours ago          Up 7 hours                    k8s_coredns_coredns-78fcd69978-sc49b_kube-system_5bf26f98-4b52-4e5b-8dbf-ee5adca62771_2
430216e1dc87   8d147537fb7d                         "/coredns -conf /etc…"   7 hours ago          Up 7 hours                    k8s_coredns_coredns-78fcd69978-xhwnz_kube-system_6f19c380-8277-4bae-a972-7a1856305d19_2
2328fe28f628   8f8fdd6672d4                         "/usr/local/bin/kube…"   7 hours ago          Up 7 hours                    k8s_kube-proxy_kube-proxy-2494n_kube-system_b1ba30bc-f010-47d6-9e15-c92d1cfd1d21_2
185f06a670e7   k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago          Up 7 hours                    k8s_POD_storage-provisioner_kube-system_026bff3b-ff08-427e-9070-335e41f0a842_2
094ffbf4d103   k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago          Up 7 hours                    k8s_POD_coredns-78fcd69978-xhwnz_kube-system_6f19c380-8277-4bae-a972-7a1856305d19_2
f38718d95ce1   k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago          Up 7 hours                    k8s_POD_vpnkit-controller_kube-system_3b9c1da2-6cd8-4f58-9994-292c60ec1c30_2
7c3be6de03ca   k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago          Up 7 hours                    k8s_POD_coredns-78fcd69978-sc49b_kube-system_5bf26f98-4b52-4e5b-8dbf-ee5adca62771_2
64da36c8a371   k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago          Up 7 hours                    k8s_POD_kube-proxy-2494n_kube-system_b1ba30bc-f010-47d6-9e15-c92d1cfd1d21_2
81b781a4a6bd   004811815584                         "etcd --advertise-cl…"   7 hours ago          Up 7 hours                    k8s_etcd_etcd-docker-desktop_kube-system_e5422d6ad44165160915 3d62d1ad42f6_2
fa53e5984047   059e6cd8cf78                         "kube-apiserver --ad…"   7 hours ago          Up 7 hours                    k8s_kube-apiserver_kube-apiserver-docker-desktop_kube-system_3ca039f7ff0c547d271c37587418b712_2
08a2fd1e5e54   04185bc88e08                         "kube-controller-man…"   7 hours ago          Up 7 hours                    k8s_kube-controller-manager_kube-controller-manager-docker-desktop_kube-system_1fe1f00ef31eb1efe8731cc2f5b776aa_3
9efb2029ccfea   935d8fdc2d52                         "kube-scheduler --au…"   7 hours ago          Up 7 hours                    k8s_kube-scheduler_kube-scheduler-docker-desktop_kube-system_d19d89767cc69ae3010bc658c1cd2e46_2
8efb5414ae38   k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago          Up 7 hours                    k8s_POD_etcd-docker-desktop_kube-system_e5422d6ad44165160915 3d62d1ad42f6_2
66ddfbac9d10   k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago          Up 7 hours                    k8s_POD_kube-apiserver-docker-desktop_kube-system_3ca039f7ff0c547d271c37587418b712_2
c1d121560a2    k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago          Up 7 hours                    k8s_POD_kube-controller-manager-docker-desktop_kube-system_1fe1f00ef31eb1efe8731cc2f5b776aa_2
5ce6bf1169eb   k8s.gcr.io/pause:3.5                 "/pause"                 7 hours ago          Up 7 hours                    k8s_POD_kube-scheduler-docker-desktop_kube-system_d19d89767cc69ae3010bc658c1cd2e46_2

C:\Users\Admin>
```

Practical No 4

Aim : Working with Docker

Commands and its output :

Step 1:

- Create a account in the docker hub. Remember the username and password of the account

Step 2 :

- Now to go https://labs.play-with-docker.com/ and click on **Start** button.
- Click on **Add New Instance**. You will see the editor open in the right pane. Give the commands in the editor

Step 3 :

- To check the version of the docker
  $ docker – version

```
[node1] (local) root@192.168.0.18 ~
$ docker --version
Docker version 20.10.0, build 7287ab3
[node1] (local) root@192.168.0.18 ~
$
```

- To pull the readymade image
  $ docker pull hello-world

```
$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:80f31da1ac7b312ba29d65080fddf797dd76acfb870e677f390d5acba9741b17
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
[node1] (local) root@192.168.0.18 ~
$
```

- To check the images in docker
  $ docker images

```
$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
hello-world   latest    feb5d9fea6a5   7 months ago    13.3kB
[node1] (local) root@192.168.0.18 ~
```

Part 1: To pull and Push images in docker

Step 4 :

- Open the new tab in the browser and login to hub.docker.com
- Click on **Repositories** and then click on **Create Repositories**

- Give the name of the repository as "**repo1**" and in description add "**My first repository**"
- Make visibility as **Private**
- And now click on **Create** button and check whether the repository is created or not.

Step 5 :

- Now come to the https://labs.play-with-docker.com/ and give the following command
- Login into docker account
  $ docker login -username= your_user_name
  password:

```
[node1] (local) root@192.168.0.18 ~
$ docker login --username=vishwakarma1919
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

Note: Give your username and password that you have used to login to hub.docker.com

- To tag an image in docker
  $ docker tag <image id> <username>/repo1: firsttry

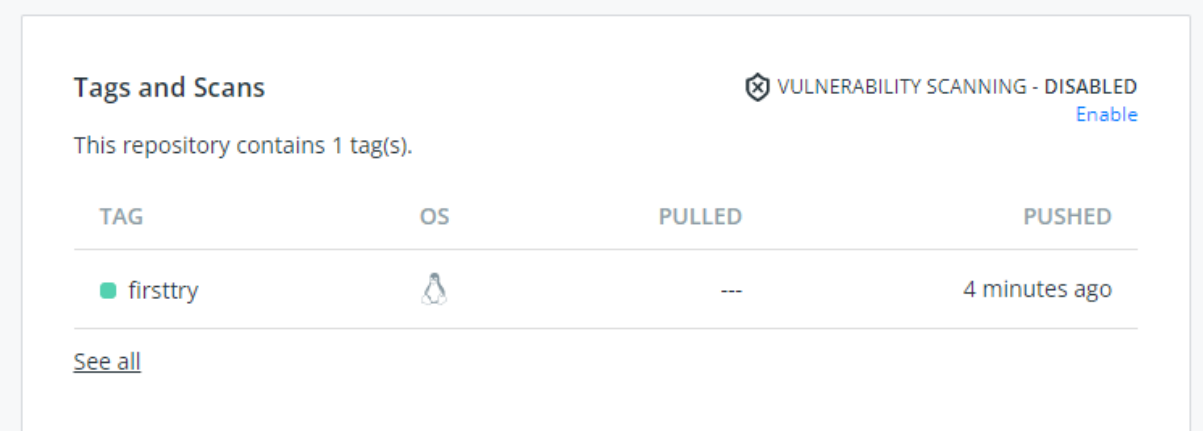
```
[node1] (local) root@192.168.0.18 ~
$ docker tag feb5d9fea6a5 vishwakarma1919/repo1:firsttry
[node1] (local) root@192.168.0.18 ~
$
```

- To push the image to docker account
  $ docker push <username>/repo1:firsttry

```
[node1] (local) root@192.168.0.18 ~
$ docker push vishwakarma1919/repo1:firsttry
The push refers to repository [docker.io/vishwakarma1919/repo1]
e07ee1baac5f: Mounted from library/hello-world
firsttry: digest: sha256:f54a58bc1aac5ea1a25d796ae155dc228b3f0e11d046ae276b39c4bf2f13d8c4 size: 525
[node1] (local) root@192.168.0.18 ~
```

Note: firsttry is tag name created above.

- Check it in hub.docker.com now in tags tab



Part 2 : Build and image and then push and run in the docker0

Step 6 :

- In https://labs.play-with-docker.com/ give the following command
  cat > Dockerfile <<EOF
  FROM busybox
  CMD echo "Hello world! This is my first Docker image."
  EOF

```
[node1] (local) root@192.168.0.18 ~
$ cat> Dockerfile <<EOF
> FROM  busybox
> CMD echo "Hello World! This Is My First Docker Image."
> EOF
```

- To build the image from docker file
  $ docker build –t <username>/repo2 .

```
[node1] (local) root@192.168.0.18 ~
$ docker build -t vishwakarma1919/repo2 .
Sending build context to Docker daemon     47MB
Step 1/2 : FROM  busybox
latest: Pulling from library/busybox
50e8d59317eb: Pull complete
Digest: sha256:d2b53584f580310186df7a2055ce3ff83cc0df6caacf1e3489bff8cf5d0af5d8
Status: Downloaded newer image for busybox:latest
 ---> 1a80408de790
Step 2/2 : CMD echo "Hello World! This Is My First Docker Image."
 ---> Running in 523badc76755
Removing intermediate container 523badc76755
 ---> 58a88ef19a6a
Successfully built 58a88ef19a6a
Successfully tagged vishwakarma1919/repo2:latest
[node1] (local) root@192.168.0.18 ~
$
```

- Check images in docker

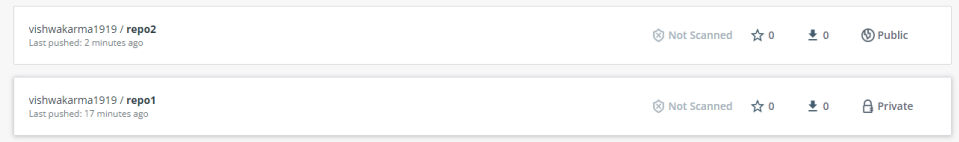
$ docker images

```
[node1] (local) root@192.168.0.18 ~
$ docker images
REPOSITORY                TAG         IMAGE ID       CREATED          SIZE
vishwakarma1919/repo2     latest      58a88ef19a6a   26 seconds ago   1.24MB
busybox                   latest      1a80408de790   4 weeks ago      1.24MB
hello-world               latest      feb5d9fea6a5   7 months ago     13.3kB
vishwakarma1919/repo1     firsttry    feb5d9fea6a5   7 months ago     13.3kB
[node1] (local) root@192.168.0.18 ~
$
```

- To push the image on the docker hub
  $ docker push <username>/repo2.

```
[node2] (local) root@192.168.0.8 ~
$ docker push vishwakarma1919/repo2
Using default tag: latest
The push refers to repository [docker.io/vishwakarma1919/repo2]
eb6b01329ebe: Mounted from library/busybox
latest: digest: sha256:4452bb83a562a0ce6a5e1fa11159957b8ad3cc62dffb6ad14b60dd4e5dd29bf3 size: 527
```

- Check it in hub.docker.com now in tags tab



- Come back to the https://labs.play-with-docker.com/ and give the below command to run the docker image
  $ docker run <username>/repo2

```
[node2] (local) root@192.168.0.8 ~
$ docker run vishwakarma1919/repo2
Hello world! This is My First Docker Image
[node2] (local) root@192.168.0.8 ~
$
```

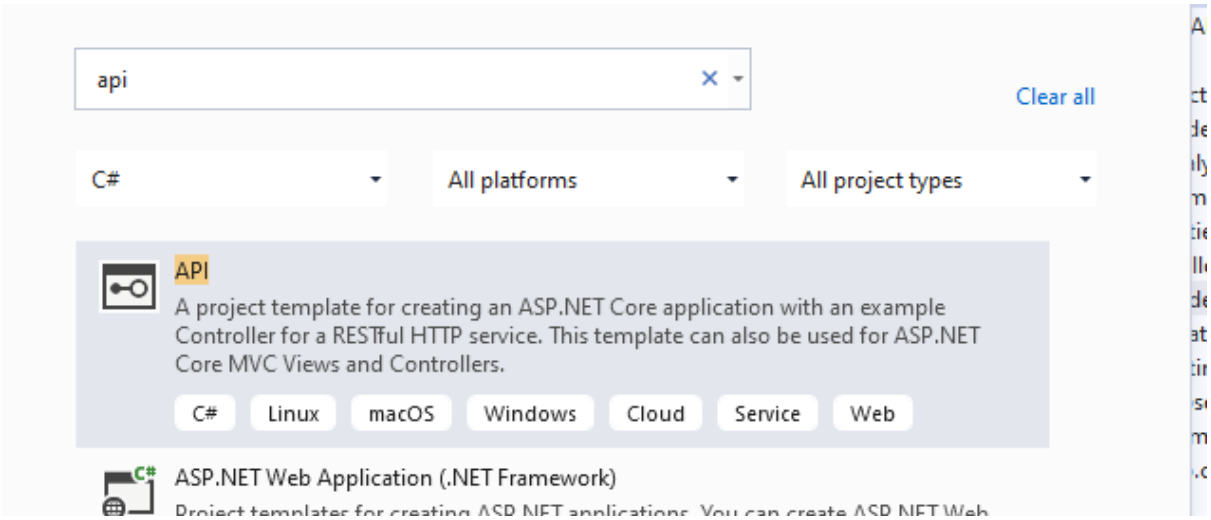
- Close the session

PRACTICAL NO 5

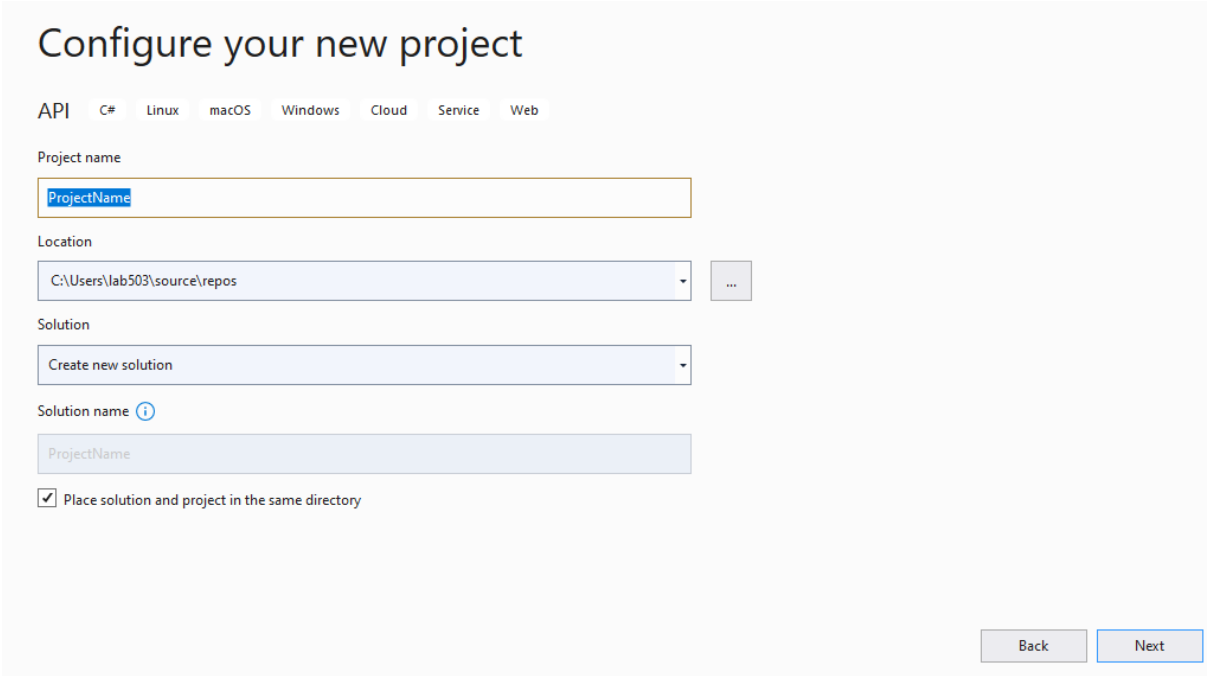
Building ASP.Net core REST API

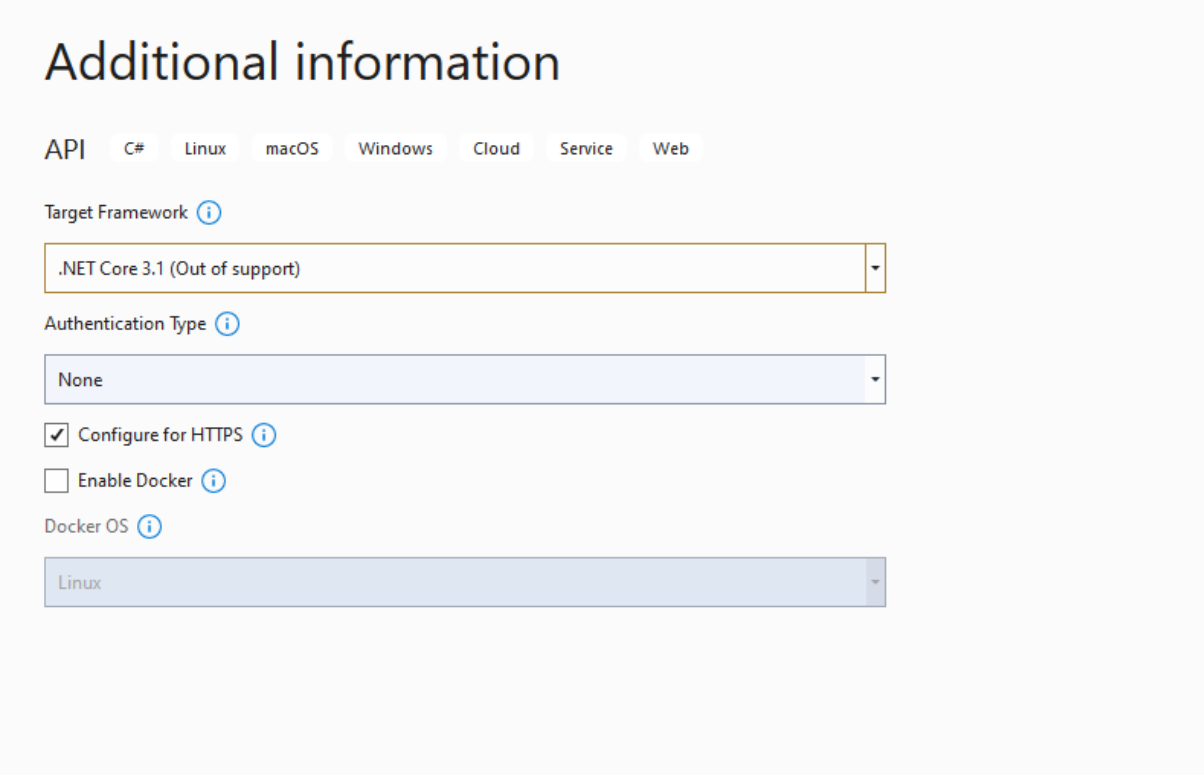
Step 1:- create new project in visual studio 2019

Step 2:- Select the Template API as mentioned below



Step 3:- Create the project with name and configures





Step 4:- You will Get all the pre-configured Code under the *Program.cs* and *Startup.cs*

Below are the Code of Startup.cs

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ProjectName
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

            // This method gets called by the runtime. Use this method to add services to          the container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddControllers();
        }
```

```
        // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.UseHttpsRedirection();

            app.UseRouting();

            app.UseAuthorization();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllers();
            });
        }
    }
}
```

Below are the Code of Program.cs

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ProjectName
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}
```

Step 5: Create Student pojo class to store the student info

Below Student.cs pojo class

```
namespace RestAPI.Controllers
{
    public class Students
    {
        public string Name  { get; set; }
        public string Rollno { get; set; }
```

```
    }
}
```

Below Student Controller class file

```
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.IO;


namespace RestAPI.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class Student : ControllerBase
    {
        //this is local database
        private static readonly List<Students> Studentlist = new List<Students>
        {
            new Students
            {
                Name = "Abdulsalam",
                Rollno = "195"
            },
            new Students
            {
                Name = "Maria",
                Rollno = "179"
            },
            new Students
            {
                Name = "imran",
                Rollno = "201"
            }

        };


        //You will get all student from here post method
        [HttpGet]
        public ActionResult<List<Students>> Get()
        {
            return Ok(Studentlist);
        }

        //You will get perticular  student from here post method
        [HttpGet]
        [Route("{Name}")]
        public ActionResult<List<Students>> Get(string Name)
        {
            var student = Studentlist.Find(student =>

            student.Name.Equals(Name, StringComparison.InvariantCultureIgnoreCase));
            if (student == null)
            {
                return NotFound();
            }
            else
            {
                return Ok(student);
            }
        }
```

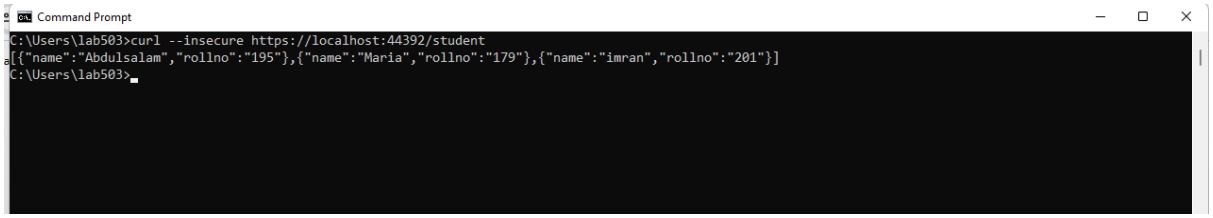
```
    //You will post perticular  student from here post method
    // post
    [HttpPost]
    public ActionResult Post(Students sentstudents)
    {
        var sendstudent = Studentlist.Find(item =>
        item.Name.Equals(sentstudents.Name,
        StringComparison.InvariantCultureIgnoreCase));
        if (sendstudent != null)
        {
            return Conflict("Cannot create the student because it already exists.");
        }
        else
        {
            Studentlist.Add(sentstudents);
            var resourceUrl = Path.Combine(Request.Path.ToString(),
            Uri.EscapeUriString(sentstudents.Name));
            return Created(resourceUrl, sentstudents);
        }
    }

    //post

    //You will add perticular student from here put method
    //put
    //
    [HttpPut]
    public ActionResult Put(Students sentstudents)
    {
        var sendstudent = Studentlist.Find(item =>
        item.Name.Equals(sentstudents.Name,
        StringComparison.InvariantCultureIgnoreCase));
        if (sendstudent == null)
        {
            return BadRequest("Cannot update a nont existing term.");
        }
        else
        {

            sendstudent.Rollno = sentstudents.Rollno;
            return Ok();
        }
    }
    //

    //delete
    // //You will Delete perticular student from here Delete method
    [HttpDelete]
    [Route("{Name}")]
    public ActionResult Delete(string Name)
    {
        var Studentnames = Studentlist.Find(item =>
        item.Name.Equals(Name,
        StringComparison.InvariantCultureIgnoreCase));
        if (Studentnames == null)
        {
            return NotFound();
        }
        else
        {
            Studentlist.Remove(Studentnames);
            return NoContent();
        }
    }

}
```

```
    //
    //
}
```

All other configurations and file will remain same

Output

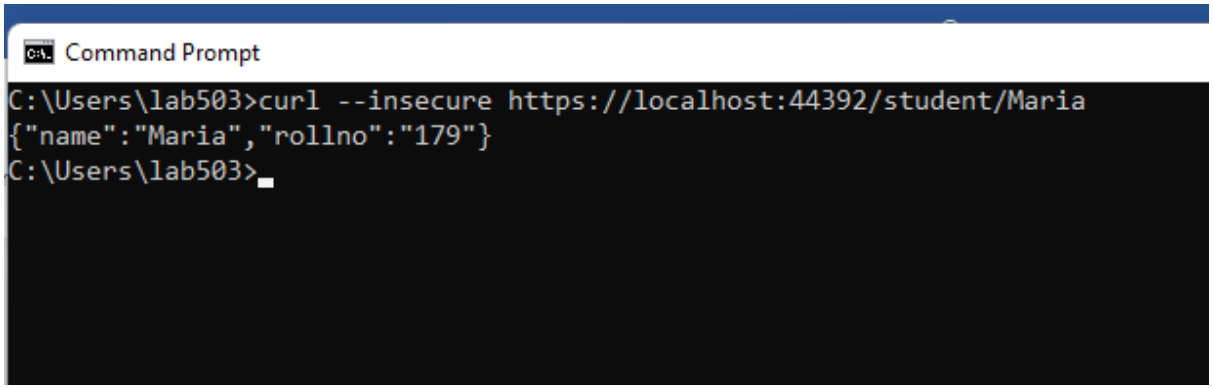
Get method



Get method

Particular Student

In this case student name *Maria*

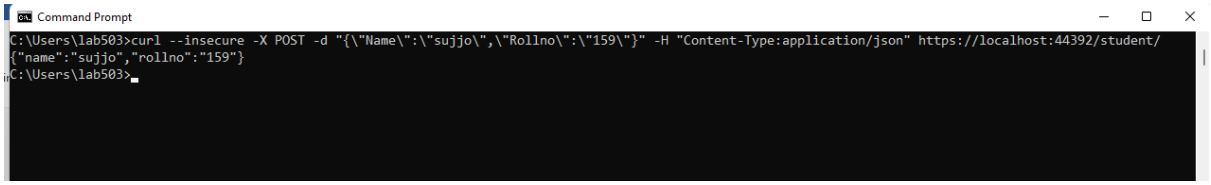


Post method

Particular Student

In this case student name *Sujjo*

curl --insecure -X POST -d "{\"Name\":\"sujjo\",\"Rollno\":\"159\"}" -H "Content-Type:application/json" https://localhost:44392/student/

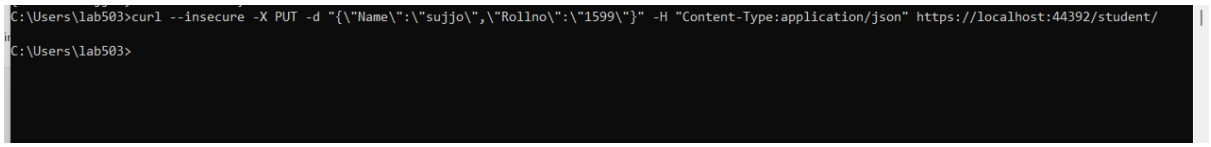


Put method

Particular Student

In this case student name *Sujjo*

curl --insecure -X PUT -d "{\"Name\":\"sujjo\",\"Rollno\":\"1599\"}" -H "Content-Type:application/json" https://localhost:44392/student/

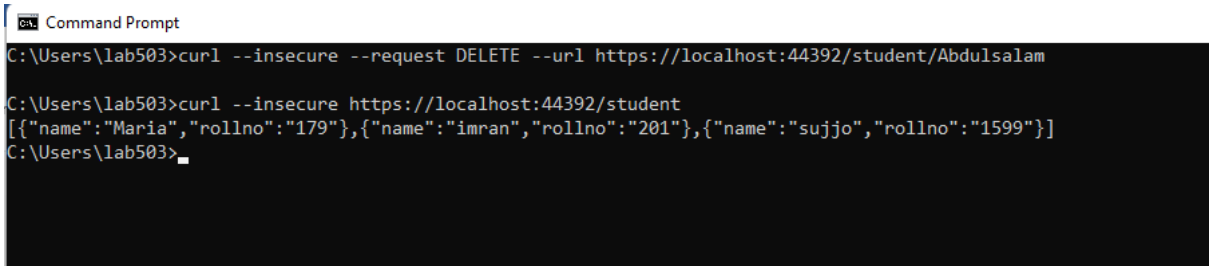


Delete

Particular Student

In this case student name *Abdulsalam*

curl --insecure --request DELETE --url https://localhost:44392/student/Abdulsalam



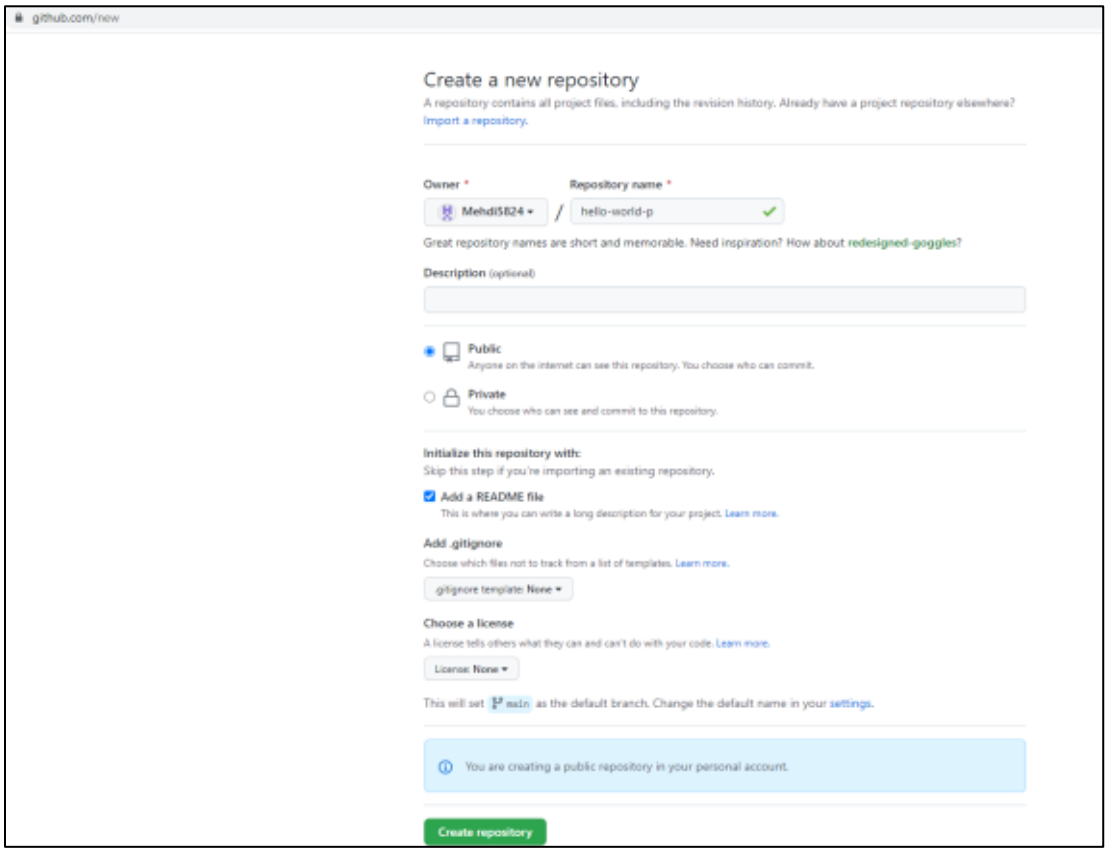
Practical No 6

Aim : Working with Circle CI for continuous integration

Steps and its output :

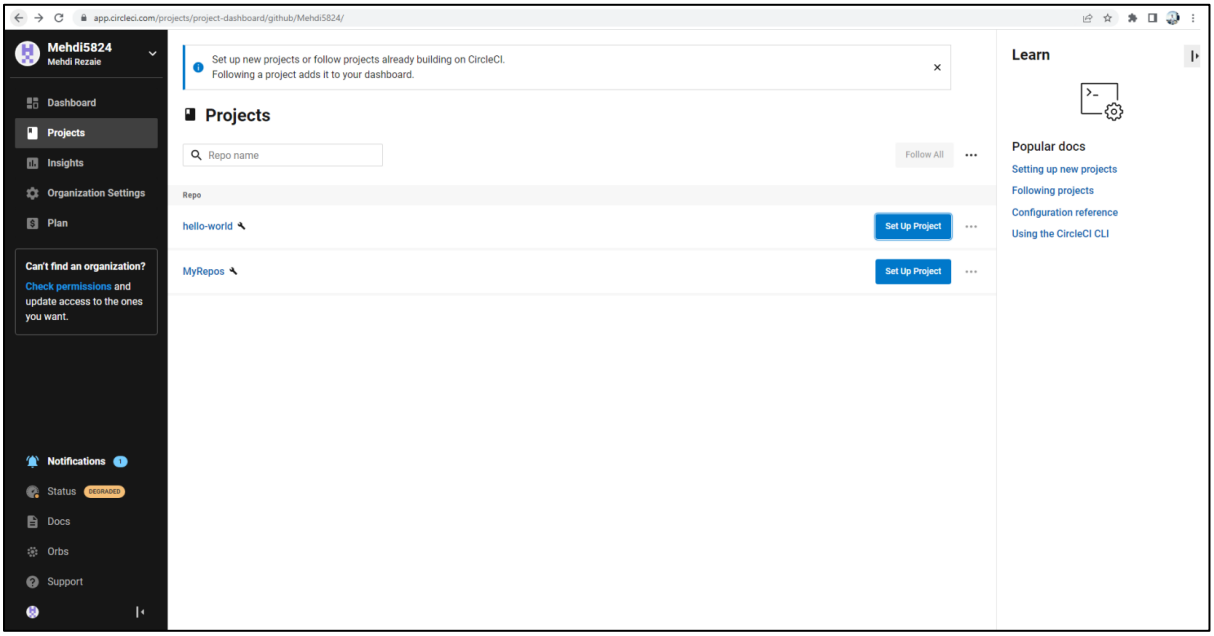
Step 1 : **Create a repository**

1. Log in to GitHub and begin the process to create a new repository.
2. Enter a name for your repository (for example, hello-world).
3. Select the option to initialize the repository with a README file.
4. Finally, click Create repository.
5. There is no need to add any source code for now.

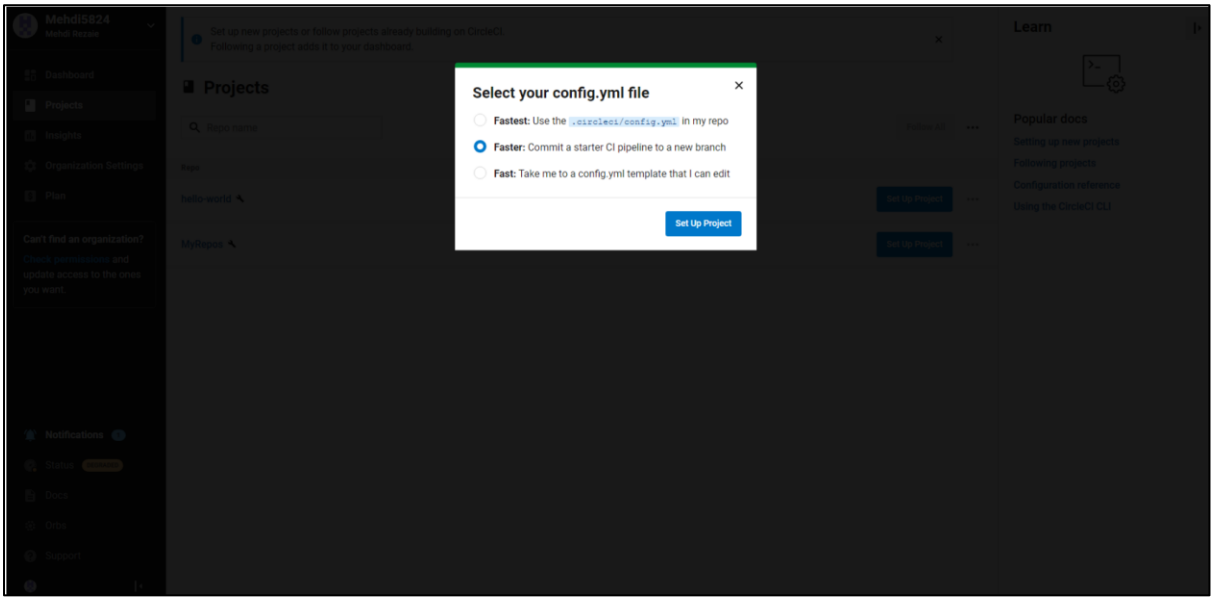


Step 2 : **Set up CircleCI**

- Login to Circle CI https://app.circleci.com/ using GitHub Login
- Navigate to the CircleCI Projects page. If you created your new repository under an organization, you will need to select the organization name.

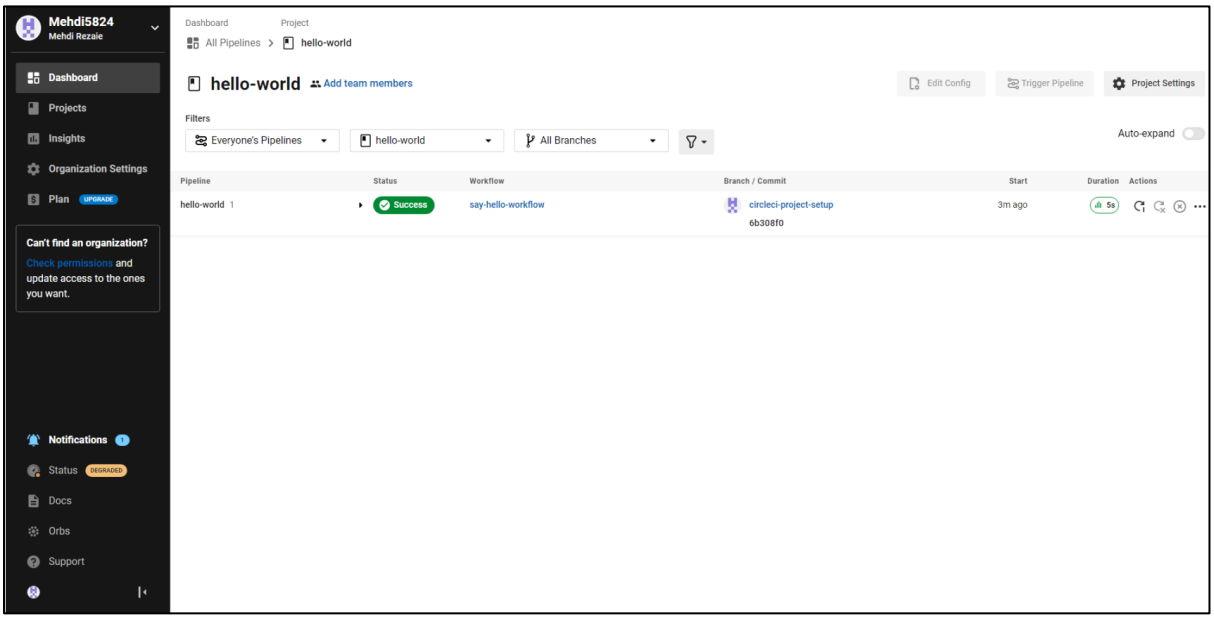


- You will be taken to the Projects dashboard. On the dashboard, select the project you want to set up (hello-world).
- Select the option to commit a starter CI pipeline to a new branch, and click Set Up Project. This will create a file ***.circleci/config.yml*** at the root of your repository on a new branch called circleci-project-setup.

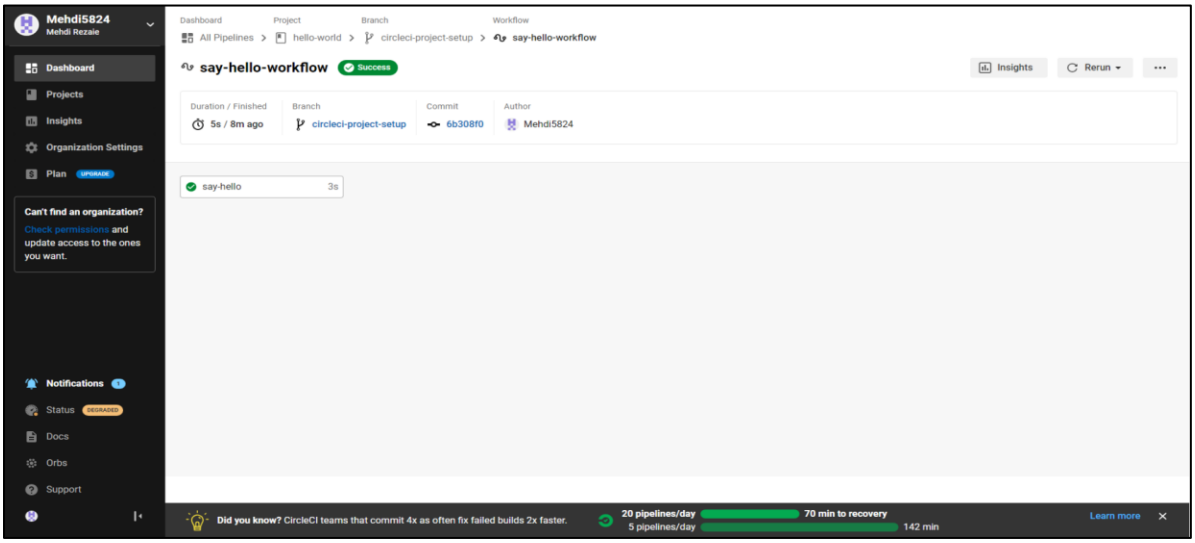


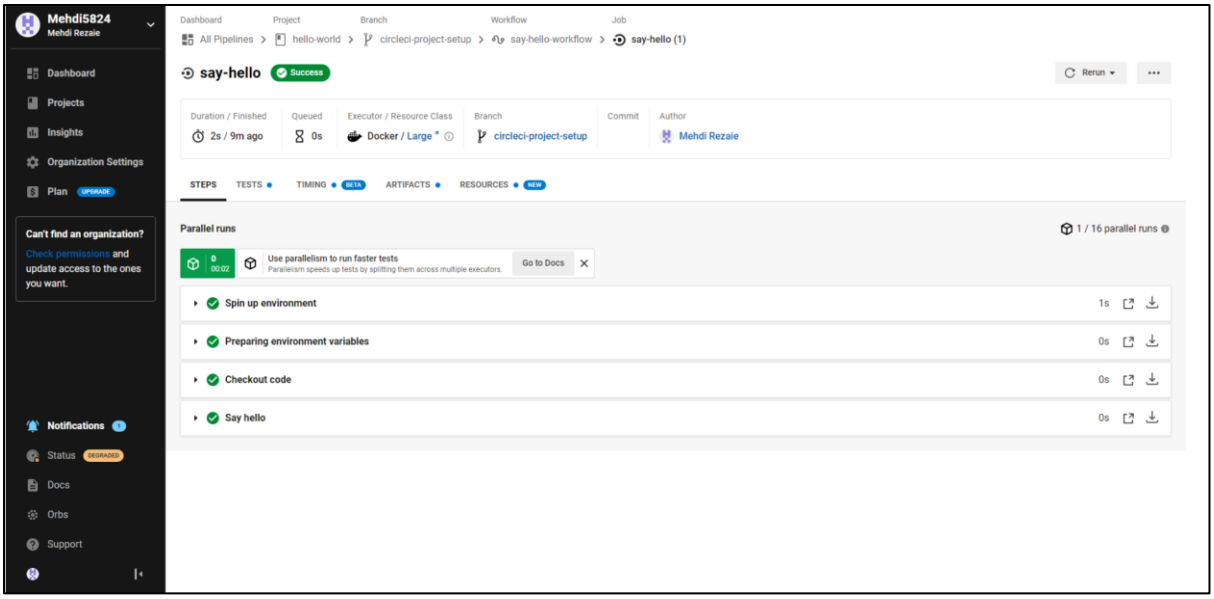
Step 3 : **Your first pipeline**

- On your project's pipeline page, click the green Success button, which brings you to the workflow that ran (say-helloworkflow).
- Within this workflow, the pipeline ran one job, called say-hello. Click say-hello to see the steps in this job:
    - Spin up environment
    - Preparing environment variables
    - Checkout code
    - Say hello
- Now select the "say-hello-workflow"

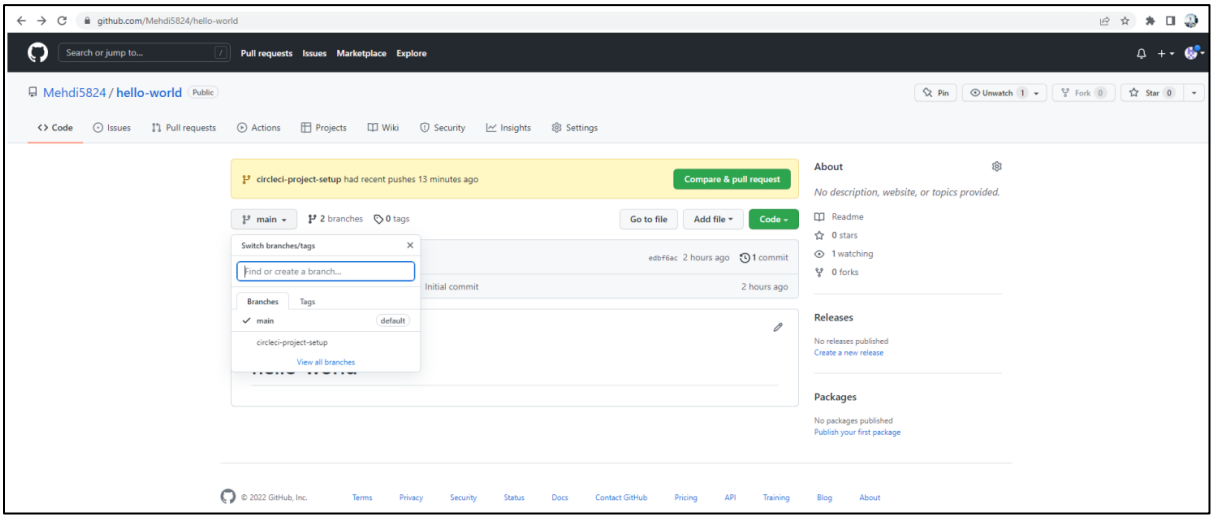


- Select "say-hello" Job with a green tick



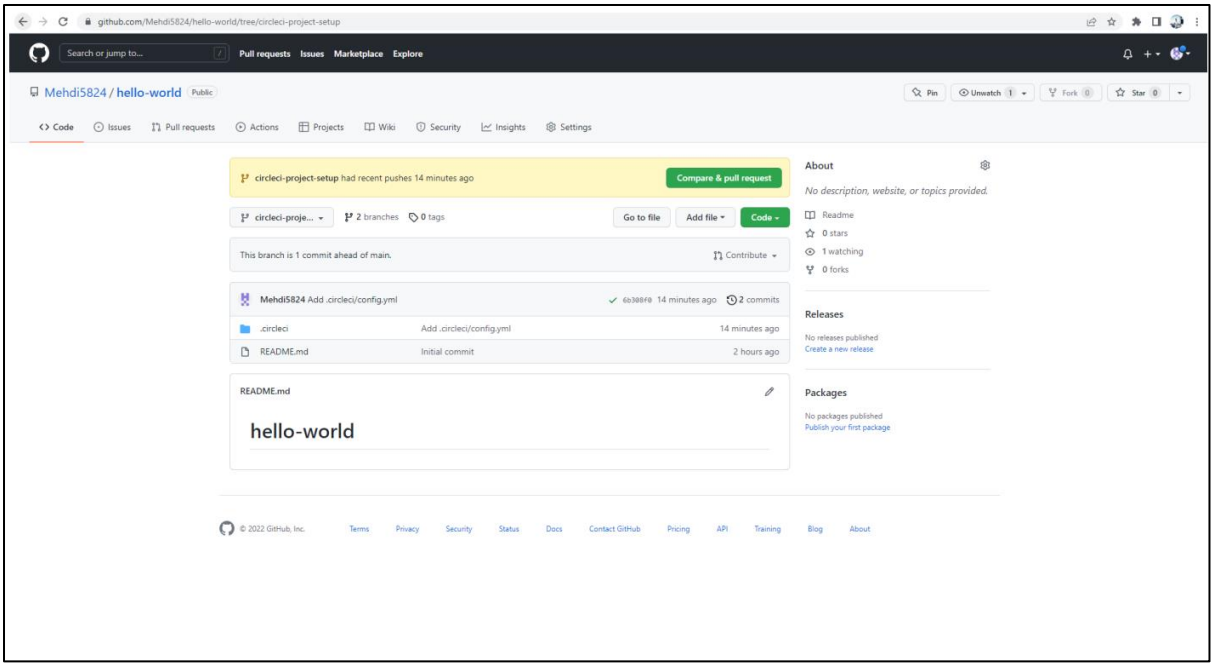


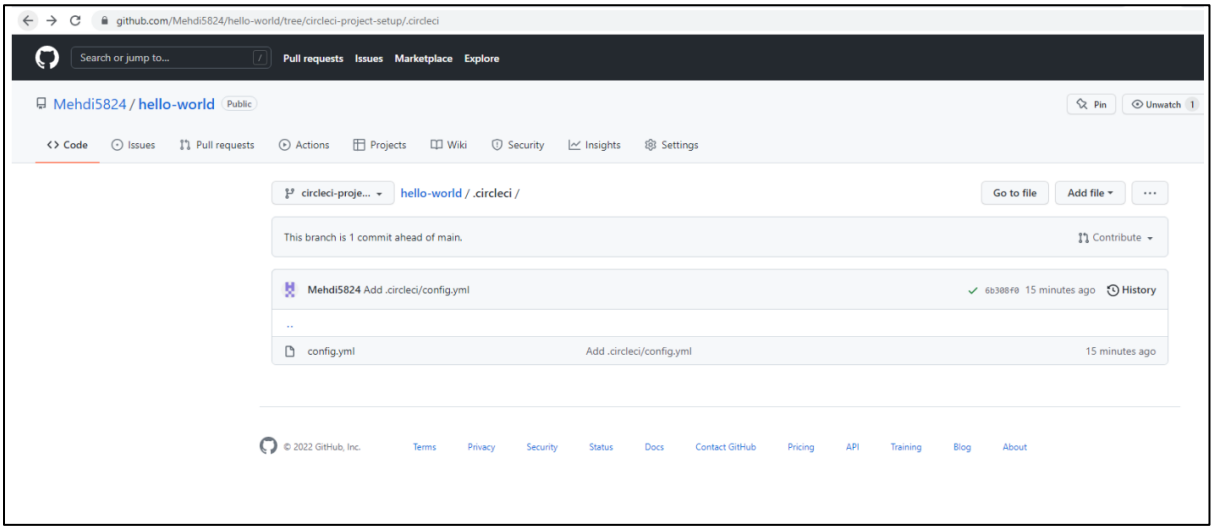
- Select Branch and option circleci-project-setup

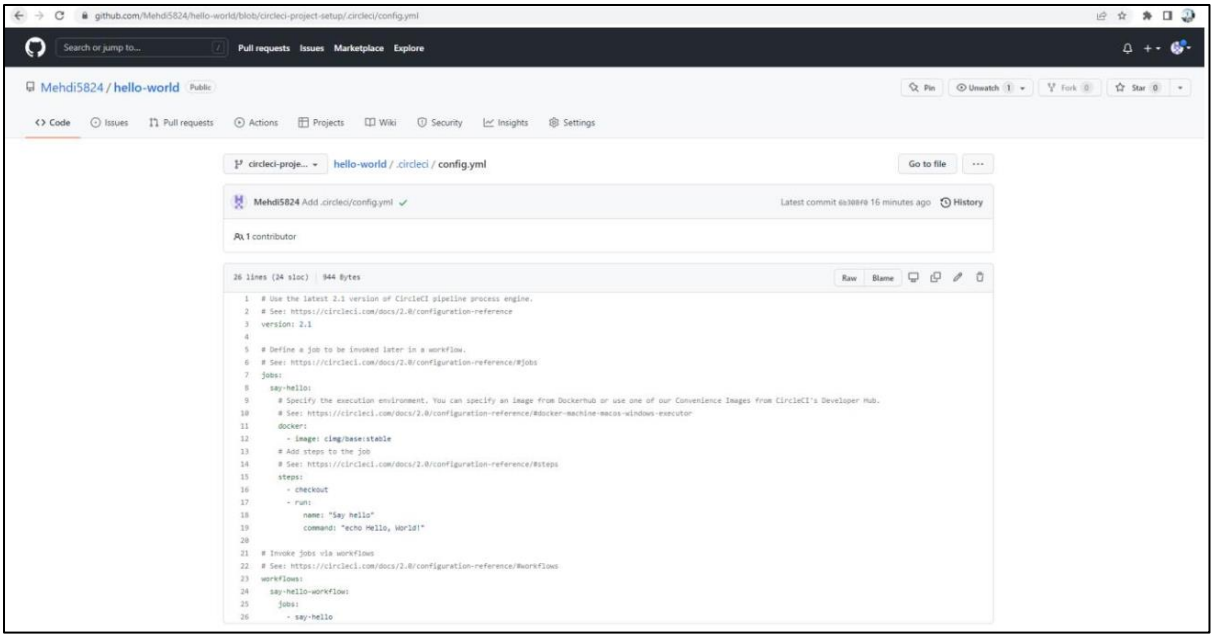


Step 4 : **Break your build**

- In this section, you will edit the .circleci/config.yml file and see what happens if a build does not complete successfully.
- It is possible to edit files directly on GitHub.


github.com/Mehdi5824/hello-world/tree/circleci-project-setup
Search or jump to...
Pull requests
Issues
Marketplace
Explore
Mehdi5824 / hello-world
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
circleci-project-setup had recent pushes 14 minutes ago
Compare & pull request
circleci-proje...
2 branches
0 tags
Go to file
Add file
Code
This branch is 1 commit ahead of main.
Contribute
Mehdi5824 Add .circleci/config.yml
6b308f0 14 minutes ago
2 commits
.circleci
Add .circleci/config.yml
14 minutes ago
README.md
Initial commit
2 hours ago
README.md
hello-world
About
No description, website, or topics provided.
Readme
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About

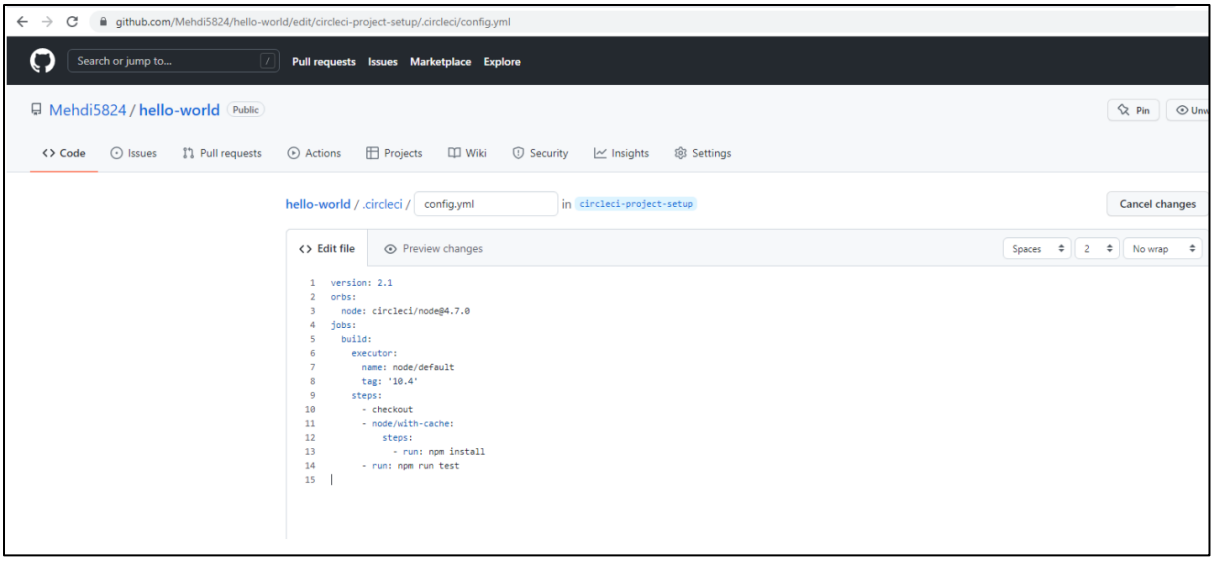

github.com/Mehdi5824/hello-world/tree/circleci-project-setup/.circleci
Search or jump to...
Pull requests
Issues
Marketplace
Explore
Mehdi5824 / hello-world
Public
Pin
Unwatch 1
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
circleci-proje...
hello-world / .circleci /
Go to file
Add file
This branch is 1 commit ahead of main.
Contribute
Mehdi5824 Add .circleci/config.yml
6b308f0 15 minutes ago
History
..
config.yml
Add .circleci/config.yml
15 minutes ago
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About



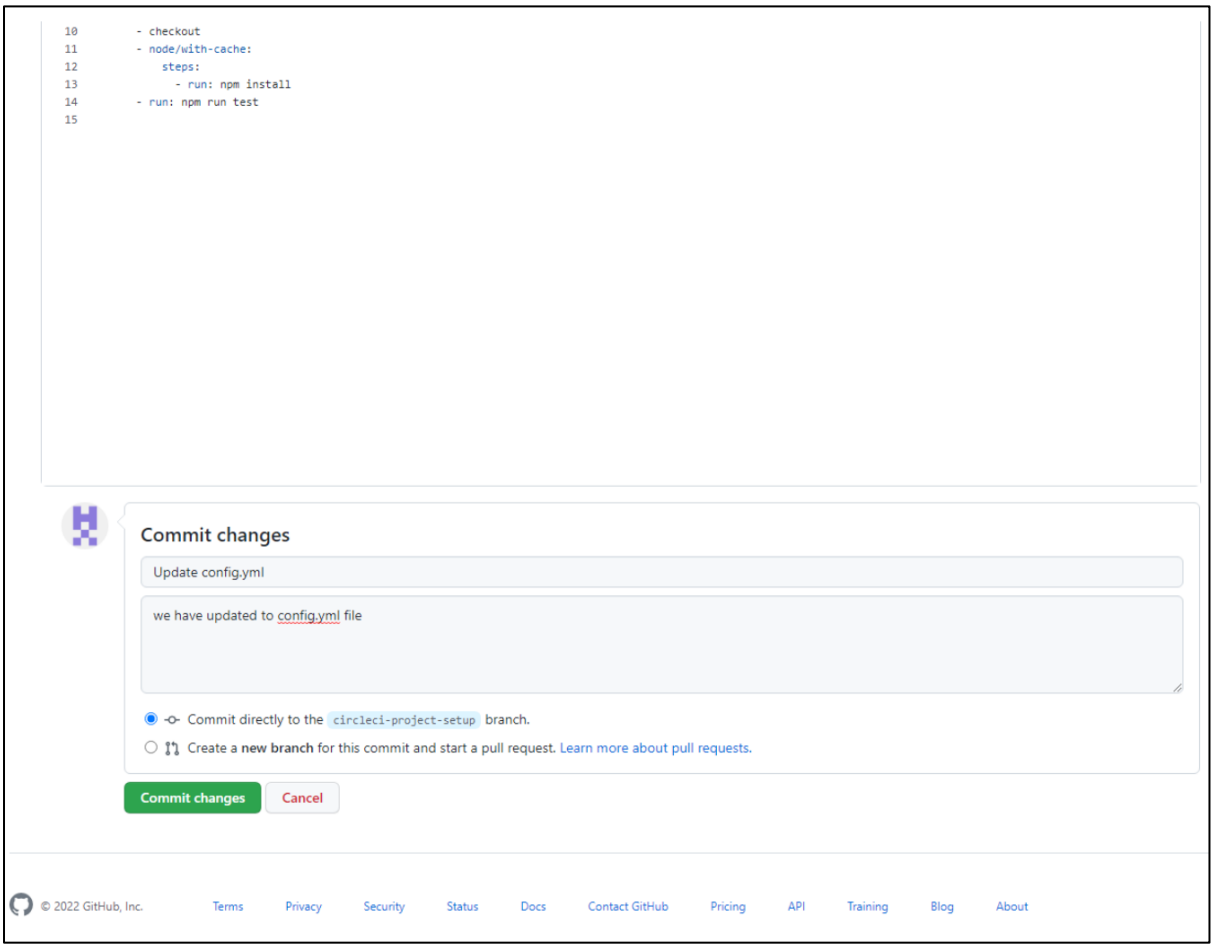
Let's use the Node orb. Replace the existing config by pasting the following code:

```yaml
version: 2.1
orbs:
  node: circleci/node@4.7.0
jobs:
  build:
    executor:
      name: node/default
      tag: '10.4'
    steps:
      - checkout
      - node/with-cache:
          steps:
            - run: npm install
      - run: npm run test
```

The GitHub file editor should look like this

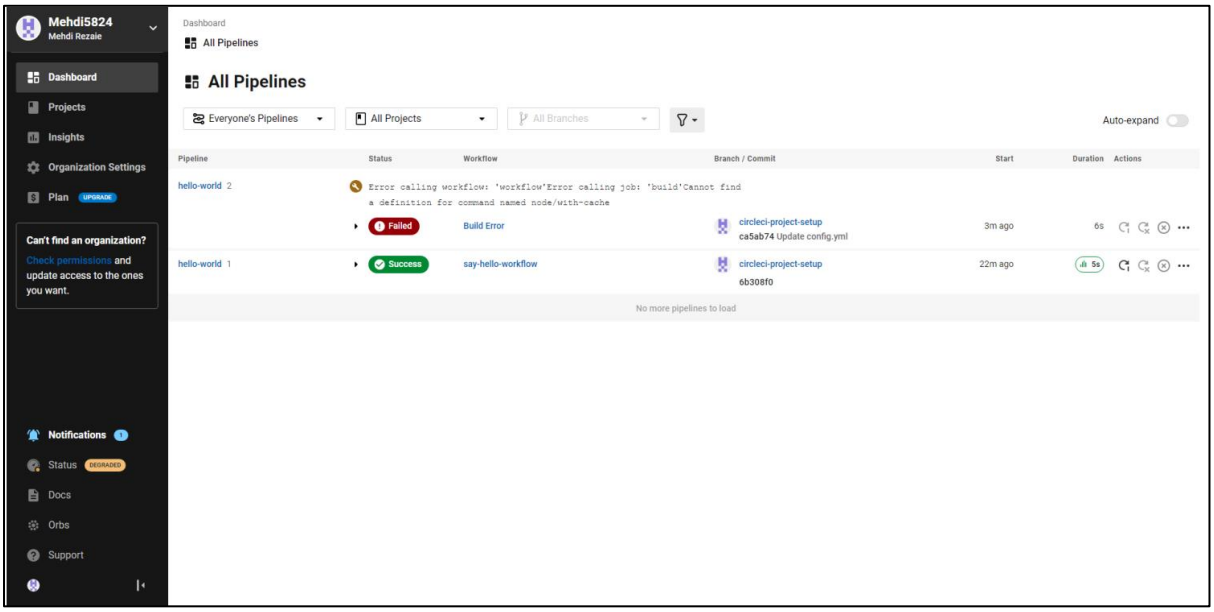


Scroll down and Commit your changes on GitHub



- After committing your changes, then return to the Projects page in CircleCI. You should see a new pipeline running… and it will fail! The Node orb runs some common Node

tasks. Because you are working with an empty repository, running npm run test, a Node script, causes the configuration to fail. To fix this, you need to set up a Node project in your repository.



Step 5 : **Use Workflows**

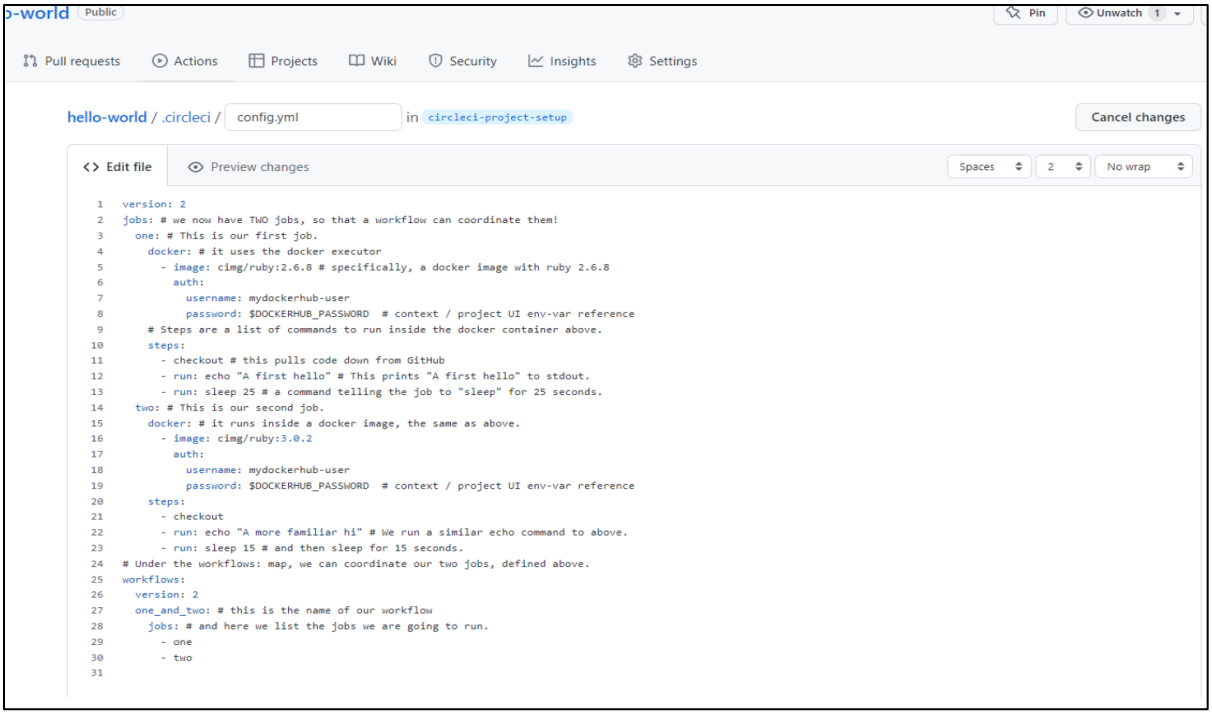
You do not have to use orbs to use CircleCI. The following example details how to create a custom configuration that also uses the workflow feature of CircleCI.

- Take a moment and read the comments in the code block below. Then, to see workflows in action, edit your .circleci/config.yml file and copy and paste the following text into it.

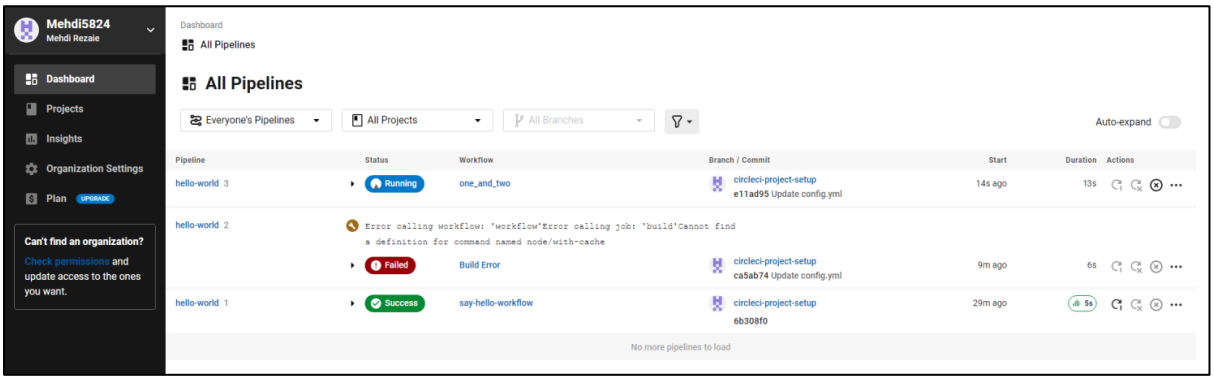
```yaml
version: 2
jobs: # we now have TWO jobs, so that a workflow can coordinate them!
  one: # This is our first job.
    docker: # it uses the docker executor
      - image: cimg/ruby:2.6.8 # specifically, a docker image with ruby 2.6.8
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD  # context / project UI env-var reference
    # Steps are a list of commands to run inside the docker container above.
    steps:
      - checkout # this pulls code down from GitHub
      - run: echo "A first hello" # This prints "A first hello" to stdout.
      - run: sleep 25 # a command telling the job to "sleep" for 25 seconds.
  two: # This is our second job.
    docker: # it runs inside a docker image, the same as above.
      - image: cimg/ruby:3.0.2
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD  # context / project UI env-var reference
    steps:
      - checkout
      - run: echo "A more familiar hi" # We run a similar echo command to above.
      - run: sleep 15 # and then sleep for 15 seconds.
# Under the workflows: map, we can coordinate our two jobs, defined above.
workflows:
  version: 2
  one_and_two: # this is the name of our workflow
    jobs: # and here we list the jobs we are going to run.
      - one
      - two
```

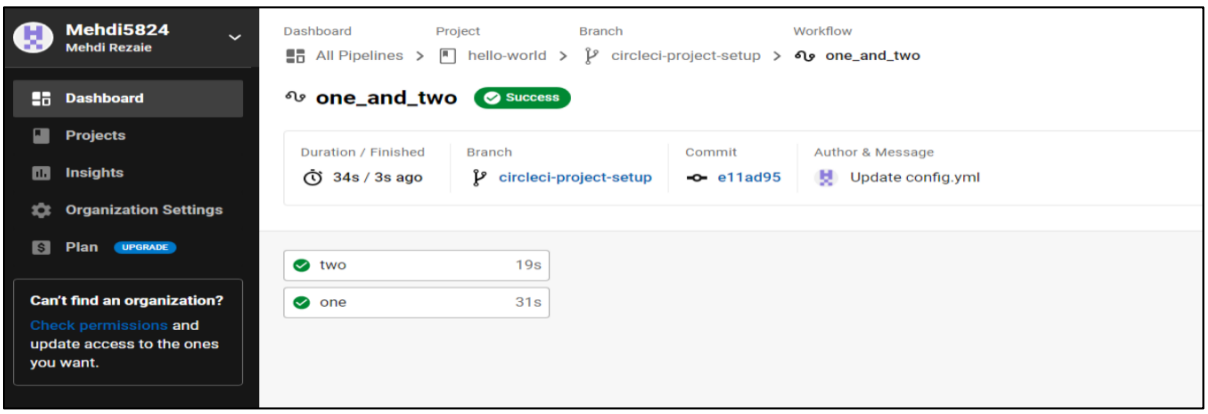
You don’t need to write the comments which are the text after #

- Commit these changes to your repository and navigate back to the CircleCI Pipelines page. You should see your pipeline running.



- Click on the running pipeline to view the workflow you have created. You should see that two jobs ran (or are currently running!) concurrently.





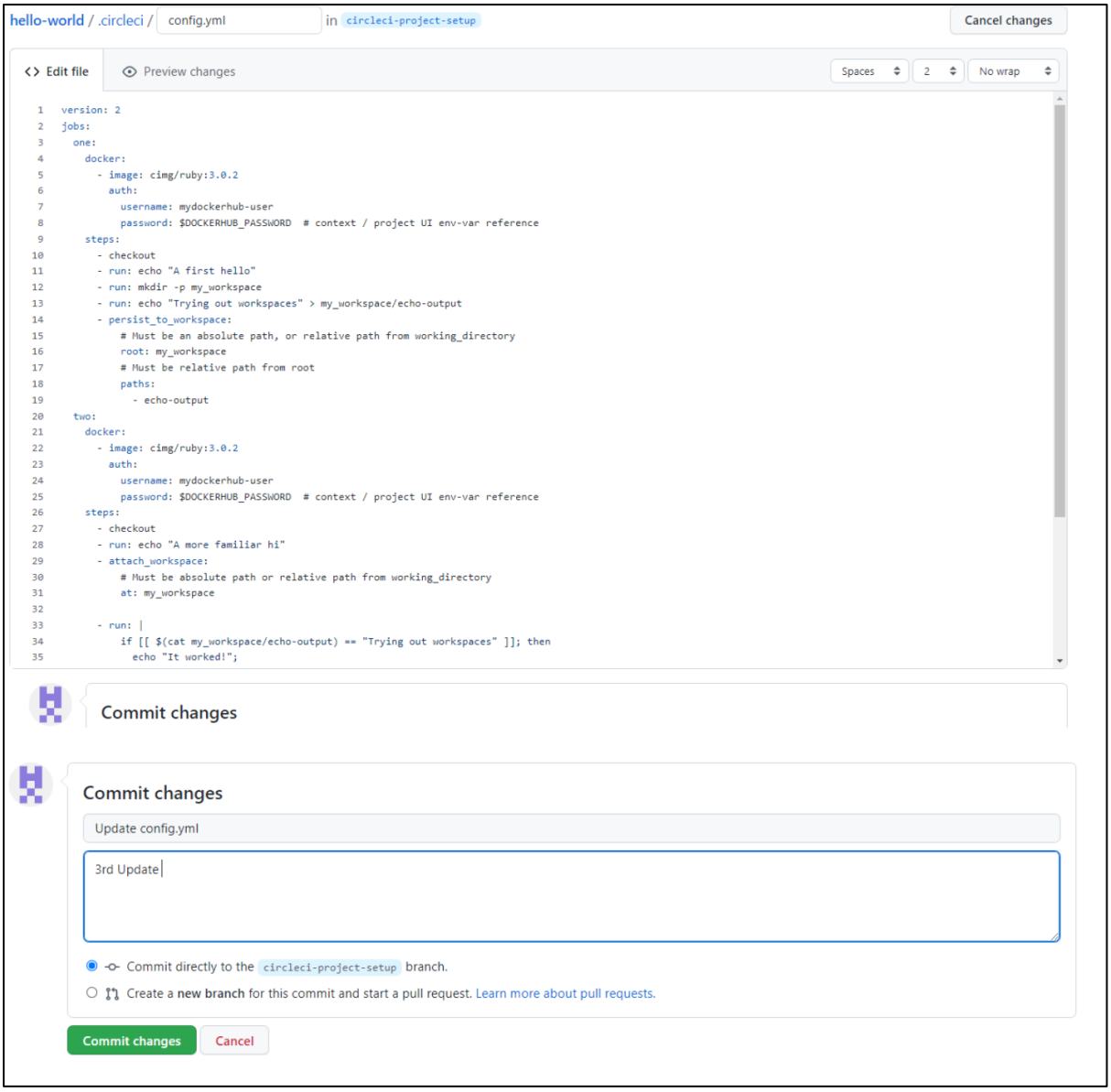
Step 6 : **Add some changes to use workspaces**

- Each workflow has an associated workspace which can be used to transfer files to downstream jobs as the workflow progresses. You can use workspaces to pass along data that is unique to this run and which is needed for downstream jobs. Try updating config.yml to the following:

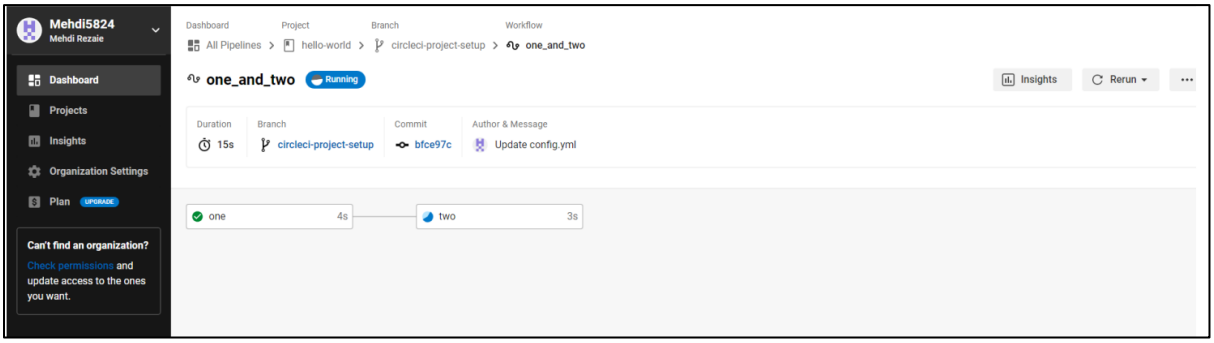
```yaml
version: 2
jobs:
  one:
    docker:
      - image: cimg/ruby:3.0.2
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD  # context / project UI env-var reference
    steps:
      - checkout
      - run: echo "A first hello"
      - run: mkdir -p my_workspace
      - run: echo "Trying out workspaces" > my_workspace/echo-output
      - persist_to_workspace:
          # Must be an absolute path, or relative path from working_directory
          root: my_workspace
          # Must be relative path from root
          paths:
            - echo-output
  two:
    docker:
      - image: cimg/ruby:3.0.2
        auth:
          username: mydockerhub-user
          password: $DOCKERHUB_PASSWORD  # context / project UI env-var reference
    steps:
      - checkout
      - run: echo "A more familiar hi"
      - attach_workspace:
          # Must be absolute path or relative path from working_directory
          at: my_workspace
```

```yaml

      - run: |
          if [[ $(cat my_workspace/echo-output) == "Trying out workspaces" ]]; then
            echo "It worked!";
          else
            echo "Nope!"; exit 1
          fi
workflows:
  version: 2
  one_and_two:
    jobs:
      - one
      - two:
          requires:
            - one
```

- Updated config.yml in GitHub file editor should be updated like this



- Finally your workflow with the jobs running should look like this



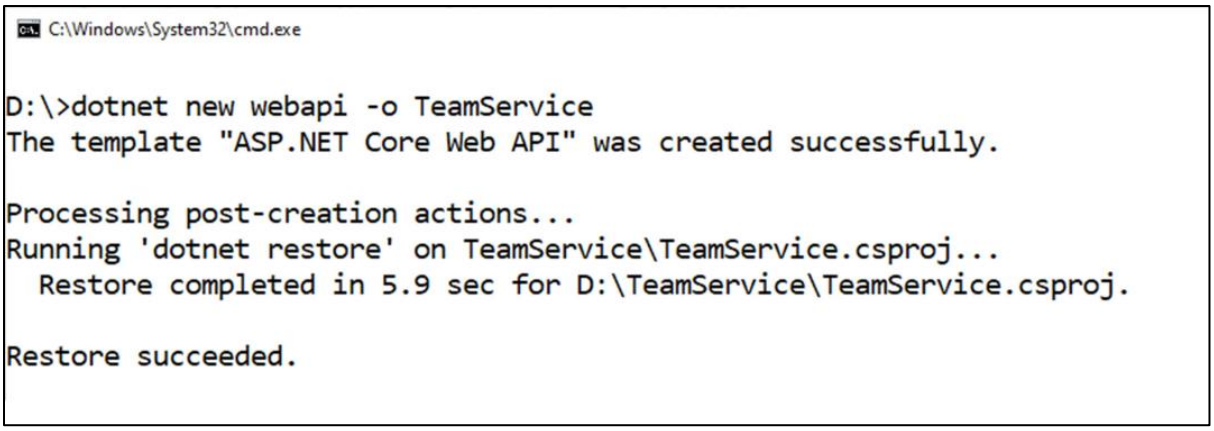
Practical No 7

Aim : Working with TeamService

Source Code :

Step 1 :

- Open command prompt and create a web api



- Remove existing weatherforecast files both model and controller files.

Step 2 :

- Add new files as follows:
- Add Member.cs to “D:\TeamService\Models” folder

```
using System;
namespace TeamService.Models
{
public class Member
{
        public Guid ID { get; set; }

        public string FirstName { get; set; }
        public string LastName { get; set; }
        public Member() { }
        public Member(Guid id) : this()
        {
        this.ID = id;
}
public Member(string firstName, string lastName, Guid id) : this(id)
{
        this.FirstName = firstName;
```

```
        this.LastName = lastName;
}
public override string ToString()
{
        return this.LastName;
}
        }
}
```

- Add Team.cs to “D:\TeamService\Models” folder

```
using System;
using System.Collections.Generic;
namespace TeamService.Models
{
        public class Team
        {
        public string Name { get; set; }
        public Guid ID { get; set; }
public ICollection<Member> Members { get; set; }public Team()
        {
        this.Members = new List<Member>();
        }
        public Team(string name) : this()
        {
        this.Name = name;
        }
        public Team(string name, Guid id) : this(name)
        {
        this.ID = id;
        }
        public override string ToString()
        {
        return this.Name;
        }
        }
}
```

- add TeamsController.cs file to “D:\TeamService\Controllers” folder

```
using System;
using Microsoft.AspNetCore.Hosting;
```

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System.Linq;
using TeamService.Models;
using System.Threading.Tasks;
using TeamService.Persistence;
namespace TeamService
{
[Route("[controller]")]
public class TeamsController : Controller
{
        ITeamRepository repository;
        public TeamsController(ITeamRepository repo)
        {
        repository = repo;

        }
        [HttpGet]
        public virtual IActionResult GetAllTeams()
        {
        return this.Ok(repository.List());

        }
        [HttpGet("{id}")]
        public IActionResult GetTeam(Guid id)
        {
        Team team = repository.Get(id);
        if (team != null)
        {
        return this.Ok(team);
        }else
        {
        return this.NotFound();
        }
        }
        [HttpPost]
        public virtual IActionResult CreateTeam([FromBody]Team newTeam)
        {
        repository.Add(newTeam);
        return this.Created($"/teams/{newTeam.ID}", newTeam);
```

```
        }
        [HttpPut("{id}")]
        public virtual IActionResult UpdateTeam([FromBody]Team team,
Guid id)
        {
        team.ID = id;
        if(repository.Update(team) == null)
        {
        return this.NotFound();
        }
        else
        {
        return this.Ok(team);
        }
        }
        [HttpDelete("{id}")]
        public virtual IActionResult DeleteTeam(Guid id)
        {
        Team team = repository.Delete(id);
        if (team == null)
        {
        return this.NotFound();
        }
        else
        {
        return this.Ok(team.ID);
        }
        }
        }
}
```

- Add MembersController.cs file to “D:\TeamService\Controllers” folder

```
using System;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System.Linq;
using TeamService.Models;
using System.Threading.Tasks;
using TeamService.Persistence;namespace TeamService
```

```
{
[Route("/teams/{teamId}/[controller]")]
        public class MembersController : Controller
        {
        ITeamRepository repository;
        public MembersController(ITeamRepository repo)
        {
        repository = repo;
        }
        [HttpGet]
        public virtual IActionResult GetMembers(Guid teamID)
        {
        Team team = repository.Get(teamID);
        if(team == null)
        {
        return this.NotFound();
        }
        else
        {
        return this.Ok(team.Members);
        }
        }
        [HttpGet]
        [Route("/teams/{teamId}/[controller]/{memberId}")]
        public virtual IActionResult GetMember(Guid teamID, Guid
memberId)
        {
        Team team = repository.Get(teamID);
        if(team == null)

        {
        return this.NotFound();
        }
        else
        {
        var q = team.Members.Where(m => m.ID == memberId);
        if(q.Count() < 1)
        {
        return this.NotFound();
        }
        else
```

```
        {
        return this.Ok(q.First());
        }
        }
        }
        }
        [HttpPut]
        [Route("/teams/{teamId}/[controller]/{memberId}")]
        public virtual IActionResult UpdateMember([FromBody]Member
updatedMember, Guid teamID, Guid memberId)
        {
        Team team = repository.Get(teamID);
        if(team == null)
        {
        return this.NotFound();
        }
        else
        {
        var q = team.Members.Where(m => m.ID == memberId);
        if(q.Count() < 1)
        {
        return this.NotFound();
        }
        else
        {
        team.Members.Remove(q.First());
        team.Members.Add(updatedMember);
        return this.Ok();
        }
        }
        }
        [HttpPost]
        public virtual IActionResult CreateMember([FromBody]Member
newMember, Guid teamID)
        {
        Team team = repository.Get(teamID);
        if(team == null)
        {
        return this.NotFound();
        }
        else
```

```
        {
        team.Members.Add(newMember);
        var teamMember = new {TeamID = team.ID, MemberID =
newMember.ID};
return
this.Created($"/teams/{teamMember.TeamID}/[controller]/{teamMember.
MemberID}", teamMember);
        }
        }
        [HttpGet]
        [Route("/members/{memberId}/team")]
        public IActionResult GetTeamForMember(Guid memberId)
        {
        var teamId = GetTeamIdForMember(memberId);
        if (teamId != Guid.Empty)
        {
        return this.Ok(new {TeamID = teamId });
        }
        else
        {
        return this.NotFound();
        }
        }
        private Guid GetTeamIdForMember(Guid memberId)
        {
        foreach (var team in repository.List())
        {
        var member = team.Members.FirstOrDefault( m => m.ID ==
memberId);
        if (member != null)
        {
        return team.ID;
        }

        }
        return Guid.Empty;
        }
        }

}
```

Step 3 :

- Create folder “D:\TeamService\Persistence”
- Add file ITeamReposiroty.cs in “D:\TeamService\Persistence” folder

```
using System;
using System.Collections.Generic;
using TeamService.Models;
namespace TeamService.Persistence
{
    public interface ITeamRepository
    {
            IEnumerable<Team> List();
            Team Get(Guid id);
            Team Add(Team team);
            Team Update(Team team);
            Team Delete(Guid id);
    }
}
```

- Add MemoryTeamRepository.cs in “D:\TeamService\Persistence” folder

```
using System;
using System.Collections.Generic;
using System.Linq;
using TeamService;
using TeamService.Models;
namespace TeamService.Persistence
{
    public class MemoryTeamRepository : ITeamRepository
    {
            protected static ICollection<Team> teams;
            public MemoryTeamRepository()
            {
                    if(teams == null)
                    {
                            teams = new List<Team>();
                    }
            }
            public MemoryTeamRepository(ICollection<Team> teams)
            {
                    MemoryTeamRepository.teams = teams;
            }
            public IEnumerable<Team> List()
```

```
        {
            return teams;
        }
        public Team Get(Guid id)
        {
            return teams.FirstOrDefault(t => t.ID == id);
        }
        public Team Update(Team t)
        {
            Team team = this.Delete(t.ID);
            if(team != null)
            {
                team = this.Add(t);
            }
            return team;
        }
        public Team Add(Team team)
        {
            teams.Add(team);
            return team;
        }
        public Team Delete(Guid id)
        {
            var q = teams.Where(t => t.ID == id);
            Team team = null;
            if (q.Count() > 0)
            {
                team = q.First();
                teams.Remove(team);
            }
            return team;
        }
    }
}
```

Step 4 :

- Add following line to Startup.cs in public void ConfigureServices(IServiceCollection services) method

```
services.AddScoped<ITeamRepository, MemoryTeamRepository>();
```

Output:

- Open two command prompt

- Command Prompt 1: go inside folder teamservice first

```
Command Prompt - dotnet run

D:\TeamService>dotnet run
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: D:\TeamService
```

- On Command Prompt 2:

To get all teams

curl --insecure https://localhost:5001/teams

```
D:\>curl --insecure https://localhost:5001/teams
[]
```

To create new team

curl --insecure -H “Content-Type:application/json” –X POST –d “{\“id\”:\”e52baa63-d511-417e-9e54-7aab04286281\”, \”name\”:\”KC\”}” https://localhost:5001/teams

```
D:\>curl --insecure -H "Content-Type:application/json" -X POST -d "{\"id\":\"e52baa63-d511-417e-9e54-7aab
04286281\",\"name\":\"KC\"}" https://localhost:5001/teams
{"name":"KC","id":"e52baa63-d511-417e-9e54-7aab04286281","members":[]}
D:\>
```

To create one more new team

curl --insecure -H “Content-Type:application/json” –X POST –d “{\“id\”:\”e12baa63-d511-417e-9e54-7aab04286281\”, \”name\”:\”MSC Part1\”}” https://localhost:5001/teams

```
D:\>curl --insecure -H "Content-Type:application/json" -X POST -d "{\"id\":\"e12baa63-d511-417e-9e54-7aab
04286281\", \"name\":\"MSC Part1\"}"  https://localhost:5001/teams
{"name":"MSC Part1","id":"e12baa63-d511-417e-9e54-7aab04286281","members":[]}
D:\>
```

To get all teams

curl --insecure https://localhost:5001/teams

```
D:\>curl --insecure https://localhost:5001/teams
[{"name":"KC","id":"e52baa63-d511-417e-9e54-7aab04286281","members":[]},{"name":"MSC Part1","id":"e12baa6
3-d511-417e-9e54-7aab04286281","members":[]}]
D:\>
```

To get single team with team-id as parameter

curl --insecure https://localhost:5001/teams/e52baa63-d511-417e-9e54-7aab04286281

```
D:\> curl --insecure https://localhost:5001/teams/e52baa63-d511-417e-9e54-7aab04286281
{"name":"KC","id":"e52baa63-d511-417e-9e54-7aab04286281","members":[]}
D:\>
```

To update team details (change name of first team from "KC" to "KC IT DEPT")

curl --insecure -H "Content-Type:application/json" –X PUT –d "{\"id\":\"e52baa63-d511-417e-9e54- 7aab04286281\", \"name\":\"KC IT DEPT\"}" https://localhost:5001/teams/e52baa63-d511-417e-9e54-7aab04286281

```
D:\>curl --insecure -H "Content-Type:application/json" -X PUT -d "{\"id\":\"e52baa63-d511-417e-9e54-7aab0
4286281\", \"name\":\"KC IT DEPT\"}"  https://localhost:5001/teams/e52baa63-d511-417e-9e54-7aab04286281
{"name":"KC IT DEPT","id":"e52baa63-d511-417e-9e54-7aab04286281","members":[]}
D:\>
```

To delete team

curl --insecure -H "Content-Type:application/json" –X DELETE https://localhost:5001/teams/e52baa63-d511-417e-9e54-7aab04286281

```
Command Prompt

D:\>curl --insecure -H "Content-Type:application/json" -X DELETE   https://localhost:5001/teams/e52baa63-
d511-417e-9e54-7aab04286281
"e52baa63-d511-417e-9e54-7aab04286281"
D:\>
```

Confirm: with get all teams now it shows only one team (first one is deleted)

curl –insecure https://localhost:5001/teams

```
Command Prompt

D:\>curl --insecure https://localhost:5001/teams
[{"name":"MSC Part1","id":"e12baa63-d511-417e-9e54-7aab04286281","members":[]}]
D:\>
```

DEPARTMENT OF INFORMATION TECHNOLOGY

N. G. ACHARYA & D. K. MARATHE COLLEGE

(Affiliated to University of Mumbai)

MUMBAI – MAHARASHTRA - 400071

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that Poojary Manish Sudhakar bearing SeatNo:4133186 submitted journal of data science and research in computing in partial fulfillment of the requirements for the award of Degree of MASTER OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Internal Guide **Coordinator**

External Examiner

Date: **College Seal**

Practical No.1(A)

Aim : Program to calculate number of samples required for an image

In image processing and computer vision, it is often necessary to process images in smaller chunks or samples. This is because processing a large image as a whole can be computationally expensive and may not be necessary for certain tasks. By breaking an image down into smaller samples, we can reduce the processing time and focus only on the relevant parts of the image.

To calculate the number of samples required for an image, we need to know the size of the image and the desired size of each sample. The provided Python program does this by using the Pillow library to open the image and obtain its width and height. We then divide both dimensions by the desired sample size, rounding down to the nearest integer to ensure that all samples are the same size. We then multiply the resulting values together to get the total number of samples required.

For example, if we have an image with a width of 640 pixels and a height of 480 pixels, and we want to use samples that are 32x32 pixels in size, we would divide 640 by 32 and round down to get 20, and divide 480 by 32 and round down to get 15. We would then multiply these values together to get the total number of samples required: 300. This means we would need to process the image in 300 separate samples, each 32x32 pixels in size.

Code:

```
from PIL import Image


def calculate_num_samples(image_path, sample_size):
    image = Image.open(image_path)
    width, height = image.size
    num_samples = (width // sample_size) * (height // sample_size)
    return num_samples

# Example usage:
num_samples = calculate_num_samples('/content/ducati.jpg', 32)
print(f"Number of samples required: {num_samples}")
```

Input:

Output:

Number of samples required: 1980

Practical No 1(B)

Aim : Program to study the effects of reducing the spatial resolution of a digital image

Spatial Resolution

The term spatial resolution corresponds to the total number of pixels in the given image. If the number of pixels is more, then the resolution of the image is more.

Down-sampling

In the down-sampling technique, the number of pixels in the given image is reduced depending on the sampling frequency. Due to this, the resolution and size of the image decrease.

Up-sampling

The number of pixels in the down-sampled image can be increased by using up-sampling interpolation techniques. The up-sampling technique increases the resolution as well as the size of the image.

Code:

```
# Import cv2, matplotlib, numpy
import cv2
import matplotlib.pyplot as plt
import numpy as np


# Read the original image and know its type
img1 = cv2.imread('g4g.png', 0)

# Obtain the size of the original image
[m, n] = img1.shape
print('Image Shape:', m, n)

# Show original image
print('Original Image:')
plt.imshow(img1, cmap="gray")


# Down sampling
# Assign a down sampling rate
# Here we are down sampling the
```

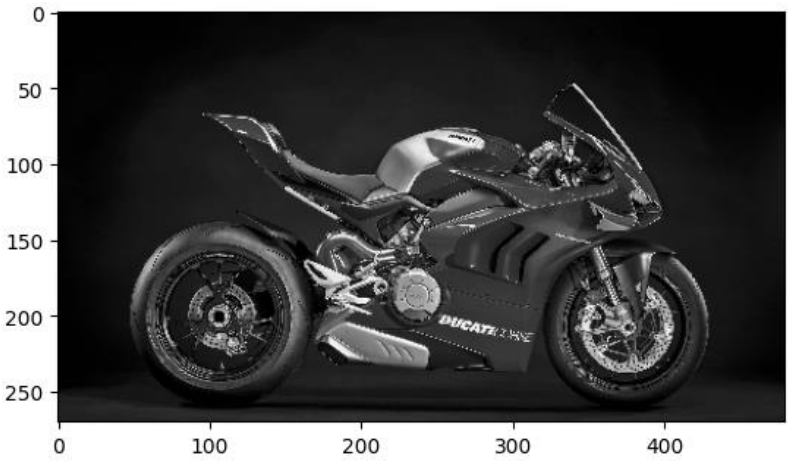
```
# image by 4
f = 4

# Create a matrix of all zeros for
# downsampled values
img2 = np.zeros((m//f, n//f), dtype=np.int)

# Assign the down sampled values from the original
# image according to the down sampling frequency.
# For example, if the down sampling rate f=2, take
# pixel values from alternate rows and columns
# and assign them in the matrix created above
for i in range(0, m, f):
        for j in range(0, n, f):
                try:

                        img2[i//f][j//f] = img1[i][j]
                except IndexError:
                        pass


# Show down sampled image
print('Down Sampled Image:')
plt.imshow(img2, cmap="gray")


# Up sampling

# Create matrix of zeros to store the upsampled image
img3 = np.zeros((m, n), dtype=np.int)
# new size
for i in range(0, m-1, f):
        for j in range(0, n-1, f):
                img3[i, j] = img2[i//f][j//f]

# Nearest neighbour interpolation-Replication
# Replicating rows

for i in range(1, m-(f-1), f):
        for j in range(0, n-(f-1)):
                img3[i:i+(f-1), j] = img3[i-1, j]

# Replicating columns
for i in range(0, m-1):
        for j in range(1, n-1, f):
                img3[i, j:j+(f-1)] = img3[i, j-1]
```

```
# Plot the up sampled image
print('Up Sampled Image:')
plt.imshow(img3, cmap="gray")
```

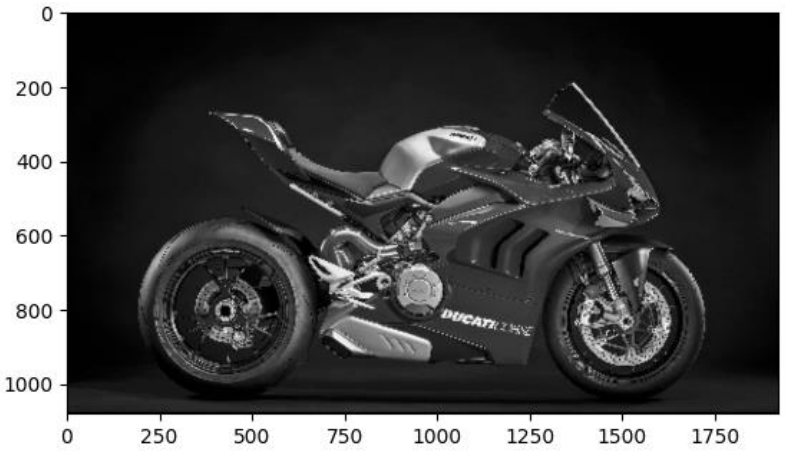
Input:

Output:

Down Sampling Image



Up Sampling Image



Practical No. 1(C)

Aim : Program to study the effects of varying the number of intensity levels in a digital image

The number of intensity levels in a digital image refers to the number of discrete brightness values that each pixel in the image can take. For example, an 8-bit grayscale image has 256 intensity levels, from 0 (black) to 255 (white).

Varying the number of intensity levels in a digital image can have several effects on the image quality. As we reduce the number of intensity levels, the image will become more pixelated and will lose detail. This is because each pixel can only represent a limited range of brightness values, leading to a loss of subtle variations in brightness.

On the other hand, increasing the number of intensity levels can lead to a smoother image with more subtle variations in brightness. However, there is a limit to how many intensity levels can be perceived by the human eye, and increasing the number beyond this limit may not result in any noticeable improvement in image quality.

In the program that varies the number of intensity levels, we can see the effects of reducing the number of intensity levels on the image quality. As the number of levels is reduced, the image becomes more pixelated and loses detail. Conversely, as the number of levels is increased, the image becomes smoother and has more subtle variations in brightness. By studying the effects of varying the number of intensity levels, we can better understand the trade-offs between image quality and file size or processing time, and choose the appropriate level for our particular application.

Code:

```
import cv2
```

```
import numpy as np
from google.colab.patches import cv2_imshow

# Load the image
img = cv2.imread("/content/water.jpg", 0)

# Display the original image
#cv2_imshow(img)

# Define the number of intensity levels to use
num_levels = [2, 4, 8, 16, 32, 64, 128]

# Loop through each number of intensity levels and display the resulting image
for n in num_levels:
    # Calculate the maximum intensity value for the new number of levels
    max_val = 256 // n

    # Apply the new number of levels to the image
```

```
    img_new = np.uint8(np.floor_divide(img, max_val) * (255 // (num_levels[-1] - 1)))

    # Display the resulting image
    cv2_imshow(img_new)
    print("\n\n")

# Wait for a key press and then close all windows
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

Output:

Practical No. 1(D)

Aim : Program to perform image averaging (image addition) for noise reduction

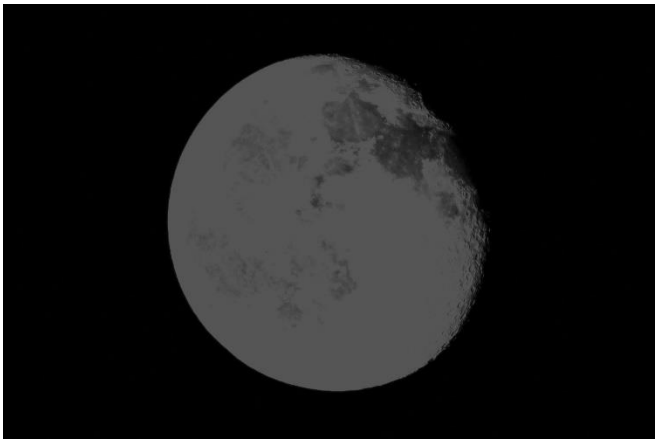
Image averaging is a technique for reducing noise in an image. It involves adding multiple images of the same scene and then dividing the sum by the number of images. The resulting image has a higher signal-to-noise ratio than the individual images, since the noise in each image is random and will cancel out to some extent when the images are added together.

Code:

```
import cv2
import numpy as np

# Load the images
img1 = cv2.imread("/content/image1.png")
img2 = cv2.imread("/content/image2.png")
img3 = cv2.imread("/content/image3.png")

# Convert the images to grayscale
gray_img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
gray_img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
gray_img3 = cv2.cvtColor(img3, cv2.COLOR_BGR2GRAY)

# Add the images together
sum_img = cv2.add(gray_img1, gray_img2)
sum_img = cv2.add(sum_img, gray_img3)

# Divide the sum by the number of images to get the average
avg_img = sum_img / 3

# Convert the average back to uint8 type
avg_img = avg_img.astype(np.uint8)

# Save the averaged image in the directory this program file is located
cv2.imwrite("averaged_image.jpg", avg_img)
```

Input:

Output:



Practical No. 1(E)

Aim: Program to compare images using subtraction for enhancing the difference between images

Image Averaging allows you to reduce the noise in the digital images through multiple exposures without damaging the details, using the fact that the digital noise is completely random and is never the same in two separate photos

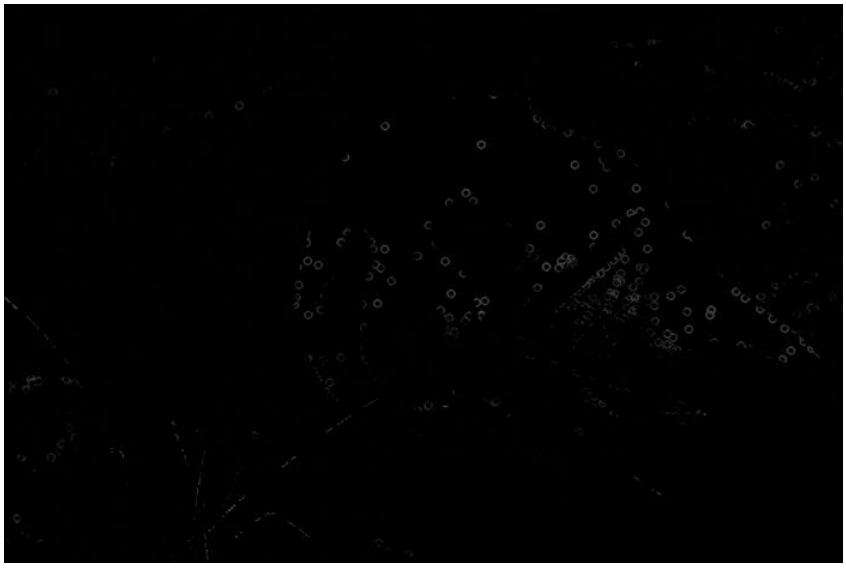
Code:

```
import cv2
```

```
from google.colab.patches import cv2_imshow
import numpy as np

# load the input images
img1 = cv2.imread('/content/1.jpg')
img2 = cv2.imread('/content/2.jpg')

# convert the images to grayscale
img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

# define the function to compute MSE between two images
def mse(img1, img2):
   h, w = img1.shape
   diff = cv2.subtract(img1, img2)
   err = np.sum(diff**2)
   mse = err/(float(h*w))
   return mse, diff

error, diff = mse(img1, img2)
print("Image matching Error between the two images:",error)

cv2_imshow(diff)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

Output:



Practical No. 1(F)

Aim: Program for Image Registration

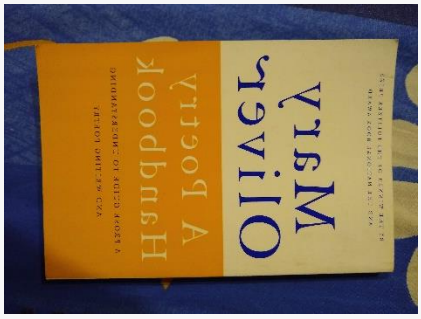
Image registration is a digital image processing technique that helps us align different images of the same scene. For instance, one may click the picture of a book from various angles. Below are a few instances that show the diversity of camera angles

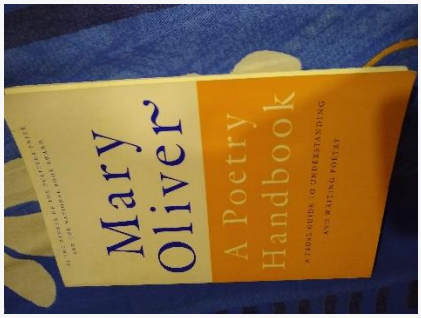
Code:

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
# Open the image files.
img1_color = cv2.imread("/content/im22.jpg") # Image to be aligned.
img2_color = cv2.imread("/content/im1.jpg") # Reference image.

# Convert to grayscale.
img1 = cv2.cvtColor(img1_color, cv2.COLOR_BGR2GRAY)
img2 = cv2.cvtColor(img2_color, cv2.COLOR_BGR2GRAY)
height, width = img2.shape

# Create ORB detector with 5000 features.
orb_detector = cv2.ORB_create(5000)

# Find keypoints and descriptors.
# The first arg is the image, second arg is the mask
# (which is not required in this case).
kp1, d1 = orb_detector.detectAndCompute(img1, None)
kp2, d2 = orb_detector.detectAndCompute(img2, None)

# Match features between the two images.
# We create a Brute Force matcher with
# Hamming distance as measurement mode.
matcher = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck = True)

# Match the two sets of descriptors.
matches = matcher.match(d1, d2)

# Take the top 90 % matches forward.
matches = matches[:int(len(matches)*0.9)]
no_of_matches = len(matches)
```

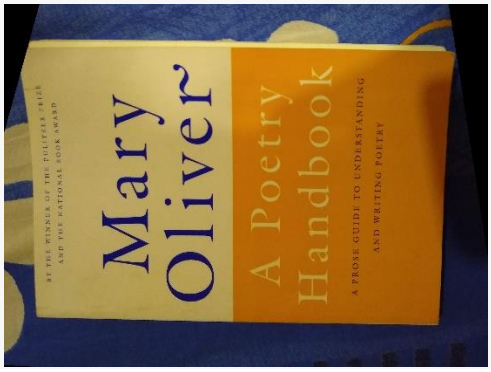
```
# Define empty matrices of shape no_of_matches * 2.
p1 = np.zeros((no_of_matches, 2))
p2 = np.zeros((no_of_matches, 2))

for i in range(len(matches)):
  p1[i, :] = kp1[matches[i].queryIdx].pt
  p2[i, :] = kp2[matches[i].trainIdx].pt

# Find the homography matrix.
homography, mask = cv2.findHomography(p1, p2, cv2.RANSAC)

# Use this matrix to transform the
# colored image wrt the reference image.
transformed_img = cv2.warpPerspective(img1_color,
        homography, (width, height))

# Save the output.
cv2.imwrite('output.jpg', transformed_img)
cv2_imshow(transformed_img)
```

Input:





Output:


Mary Oliver
A Poetry Handbook
A PROSE GUIDE TO UNDERSTANDING AND WRITING POETRY

Practical 2(A)

1) Aim : Program to perform Image Negation

The negative of an image is achieved by replacing the intensity ‘i’ in the originalimage by ‘i-1’, i.e. the darkest pixels will become the brightest and the brightest pixels will become the darkest. Image negative is produced by subtracting each pixel from the maximum intensity value.
For example in an 8-bit grayscale image, the max intensity value is 255, thuseach pixel is subtracted from 255 to produce the output image.
The transformation function used in image negative is :s =

$T(r) = (L – 1) – r$

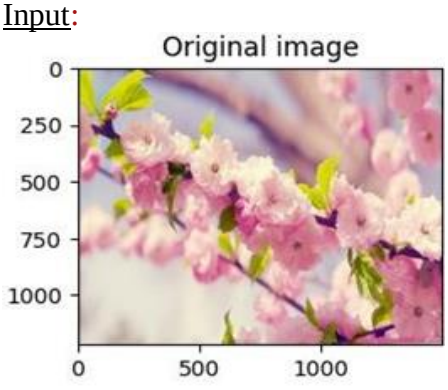
Where L - 1 is the max intensity value,s is

the output pixel value and

r is the input pixel value

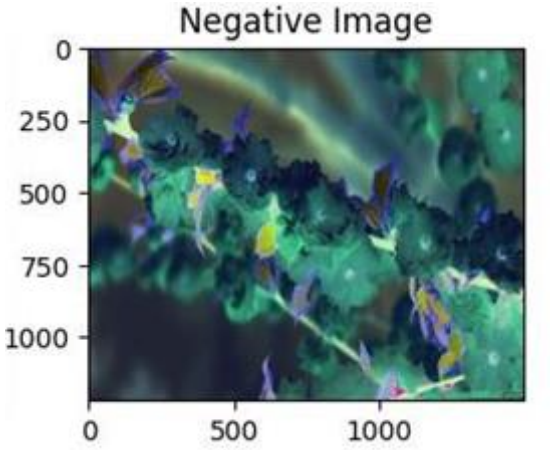
```
from pylab import *
from skimage import img_as_float

skI = imread("/content/sakura.jpg");
subplot(1, 2, 1),

imshow(skI);
title("Original image");

L = 2 ^ 8;

neg = (L - 1) - skI;
subplot(1, 2, 2),

imshow(neg);
title("Negative Image")
```

Input:



Output:

Text(0.5, 1.0, 'Negative Image')



2) Aim : Program to perform threshold of an image

Image Thresholding is an intensity transformation function in which the values of pixels below a particular threshold are reduced, and the values above that threshold are boosted. This generally results in a bilevel image at the end, wherethe image is composed of black and white pixels. Thresholding belongs to the family of point-processing techniques.

Input:

```
import cv2

# Loading the image named test.jpg
img = cv2.imread("/content/FaceDetection.jpg")

# Converting color mode to Grayscale
# as thresholding requires a single channeled image
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

cv2_imshow(img)
```

Output :

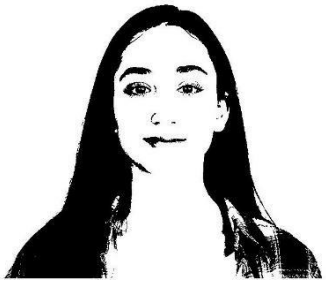
Binary Threshold

The function takes in argument a source image, threshold at which the cutoff hasto take place, maximum intensity value represented by the color space, the modeof thresholding and returns an integer value (denoting result of the operation) and an image object containing the resultant image after the processing.

Code:

```
import cv2

# Load the image in grayscale mode
img = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)

# Set the threshold value
threshold_value = 128

# Apply binary thresholding
ret, img_binary = cv2.threshold(img, threshold_value,255,cv2.THRESH_BINARY)

# Display the original and binary thresholded image
cv2.imshow('Original', img)
cv2.imshow('Binary Thresholded', img_binary)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:

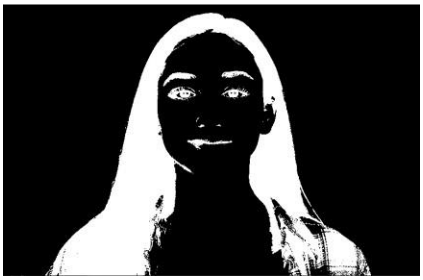


Binary-Inverse Threshold

In this, the output will be the inverse of above output i.e. white pixel will beblack and vice-versa.

```
import cv2
# Load the image in grayscale mode
img = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)

# Set the threshold value
threshold_value = 128

# Apply binary thresholding
ret, img_binary = cv2.threshold(img,
threshold_value,255,cv2.THRESH_BINARY_INV)
# Display the inverse binary thresholded image
cv2.imshow( img_binary)
cv2.waitKey(0)
cv2.destroyAllWindows()cv2.imshow('Binary Inverse Threshold', thresh)
```

Output:



Zero Thresholding

```
import cv2

# Load the image in grayscale mode
img = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)

# Set the threshold value to zero
threshold_value = 0
# Apply binary thresholding with zero threshold
ret, img_binary = cv2.threshold(img, threshold_value, 255,
cv2.THRESH_BINARY)
```

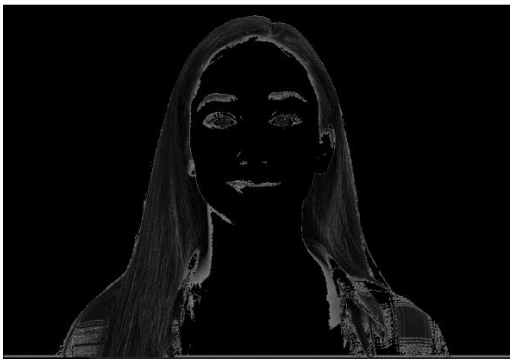
```
# Display the zero threshold binary thresholded images
cv2.imshow(img_binary)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:

Inverse Thresholding to Zero

```
import cv2

# Load the image in grayscale mode
img = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)

# Set the threshold value
threshold_value = 128

# Apply inverse thresholding to zero
ret, img_thresh = cv2.threshold(img, threshold_value, 255, cv2.THRESH_TOZERO_INV)

# Display the original and inverse thresholded images
cv2.imshow(img_thresh)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:



3) **Aim: Program to perform log transformation**

Logarithmic transformation of an image is one of the gray level image transformations. Log transformation of an image means replacing all pixel values, present in the image, with its logarithmic values. Log transformation is used for image enhancement as it expands dark pixels of the image as comparedto higher pixel values.

The formula for applying log transformation in an image is,S = c

* log (1 + r)

where,

R = input pixel value,

C = scaling constant andS

= output pixel value

Code:

```
import numpy as np
# input a number as integer

a = int(input())
print("Natural log value of the input number is", np.log(a))


# If you want base of log to be set to 2
print("Log value of the number with base 2 is", np.log2(a))


# If you want base of log to be set to 10
print("Log value of the number with base 10 is", np.log10(a))
```

Input:

Output:

4) Aim : Program to perform law transformation

Law (gamma) transformations can be mathematically expressed as $s=cr^{\gamma}$. Gamma correction is important for displaying images on a screen correctly, to prevent bleaching or darkening of images when viewed from different types of monitors with different display settings. This is done because our eyes perceiveimages in a gamma-shaped curve, whereas cameras capture images in a linear fashion. Below is the Python code to apply gamma correction.

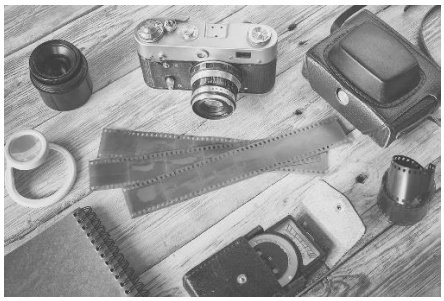
```
import cv2

import numpy as np


# Open the image.

img = cv2.imread('sample.jpg')
# Trying 4 gamma values.

for gamma in [0.1, 0.5, 1.2, 2.2]:

# Apply gamma correction.

gamma_corrected = np.array(255*(img / 255) ** gamma, dtype = 'uint8')

# Save edited images.

cv2.imwrite('gamma_transformed'+str(gamma)+'.jpg',gamma_corrected)
```

Input:

Output:

Gamma = 0.1:

Gamma = 0.5:



Gamma = 1.2:

Gamma = 2.2:

5) Aim : Program to perform Piecewise Linear Transformation function

Piecewise linear transformations are techniques used in image processing to enhance or modify the appearance of an image by applying a linear function to the pixel values of an image. There are several types of piecewise linear transformations, including:

a. Contrast Stretching:

Contrast stretching is a technique that increases the dynamic range of an image by stretching the intensity values of the image to cover the full range of pixel values. This is achieved by applying a linear function to the pixel values of the image. In this technique, the darkest pixel value in the image is mapped to 0, and the brightest pixel value is mapped to 255 (for an 8-bit image). The other pixel values are then linearly mapped between these two values. This technique can be used to enhance the contrast of an image and improve its visual appearance.

Code:

```
import cv2

# Load the image
img = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)

# Calculate the minimum and maximum pixel values
min_val, max_val, _, _ = cv2.minMaxLoc(img)

# Apply contrast stretching
img_stretched = cv2.convertScaleAbs(img, alpha=255/(max_val-min_val), beta=-
255*min_val/(max_val-min_val))

# Display the original and stretched images
#cv2.imshow('Original', img)
cv2.imshow(img_stretched)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

Output:

b. Gray-level slicing with and without background:

Gray-level slicing is a technique used to highlight specific regions of an image based on their intensity values. In this technique, a range of intensity values is selected, and all pixels within that range are highlighted while pixels outside that range are set to a specified background color. This technique can be used to isolate specific features or objects in an image.

In gray-level slicing without background, the selected range of intensity values is highlighted, and all other pixels are set to black. In gray-level slicing with background, the selected range of intensity values is highlighted, and all other pixels are set to a specified background color.

Code:

```
import cv2
import numpy as np

# Load the image
img = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)

# Define the lower and upper bounds of the gray-level slice
lower = 50
```

```
upper = 150

# Apply gray-level slicing without background
img_slice_no_bg = np.where((img >= lower) & (img <= upper), 255, 0)

# Apply gray-level slicing with background
img_slice_bg = np.where((img >= lower) & (img <= upper), 255, 100)

# Display the original and sliced images
#cv2.imshow('Original', img)
#No background Image
cv2.imshow('Sliced (no background)', img_slice_no_bg)
#With background Image
cv2.imshow('Sliced (with background)', img_slice_bg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

Output:

c. Bit-plane slicing:

Bit-plane slicing is a technique used to separate the bit-planes of an image. In this technique, each pixel value in an image is represented as a binary number. Bit-plane slicing involves separating each bit of the binary number into a separate image, resulting in a set of binary images that represent different levels of detail in the original image. The least significant bit-plane contains the coarsest information, while the most significant bit-plane contains the finest information. This technique can be used for image compression and analysis, as well as for visualizing the different levels of detail in an image.

Code:

```
import cv2
import numpy as np

# Load the image
img = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)

# Convert the image to binary
img_bin = cv2.threshold(img, 128, 255, cv2.THRESH_BINARY)[1]

# Create a list of the bit-planes
bit_planes = [cv2.bitwise_and(img_bin, 2**i) for i in range(8)]

# Display the original and bit-plane images
#cv2.imshow('Original', img)
for i in range(8):
    print(“f'Bit-Plane {i}'”)
    cv2.imshow(bit_planes[i])
    print(‘\n\n’)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

Output:

Bit-Plane 0

Bit-Plane 1

Bit-Plane 2

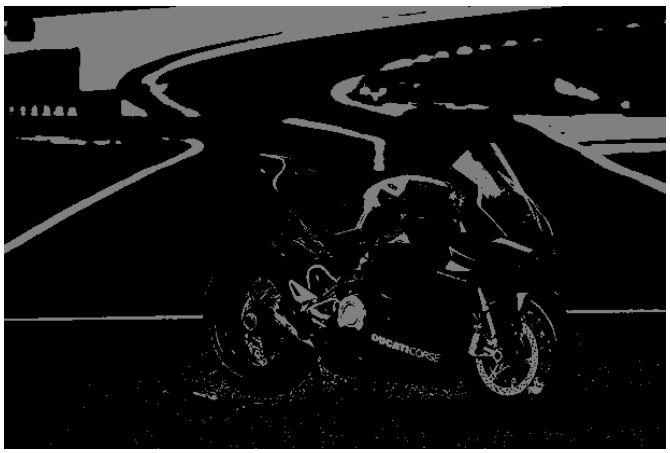
Bit-Plane 3

Bit-Plane 4

Bit-Plane 5

Bit-Plane 6

Bit-Plane 7



Pratical 2B

1) Aim: Program to plot the histogram of an image and categorize.

Histogram is considered as a graph or plot which is related to frequency of pixels in an Gray Scale Image with pixel values (ranging from 0 to 255). Grayscale image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information where pixel value varies from 0 to 255. Images of this sort, also known as black-and-white, are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest where Pixel can be considered as a every point in an image.

Code :

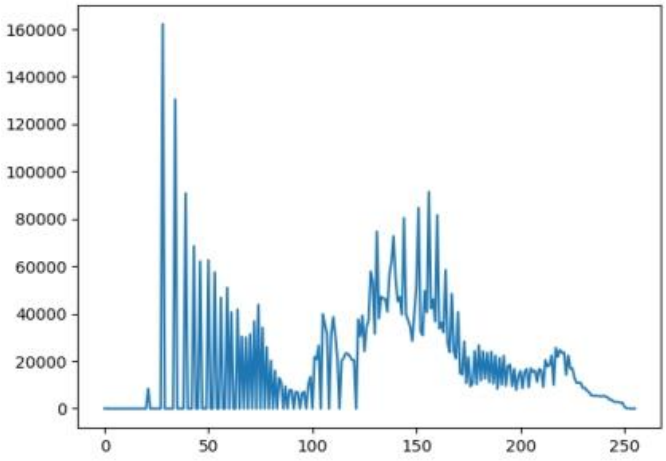
```
# importing required libraries of opencv

import cv2

# importing library for plotting

from matplotlib import pyplot as plt

# reads an input image

img = cv2.imread('/content/exe.png',0)


# find frequency of pixels in range 0-255

histr = cv2.calcHist([img],[0],None,[256],[0,256])

# show the plotting graph of an image

plt.plot(histr)

plt.show()
```

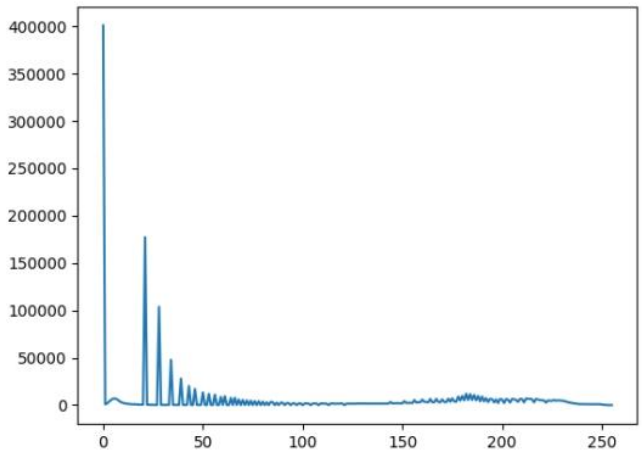
```
#/*****************************************/
# importing required libraries of opencv
import cv2
# importing library for plotting
from matplotlib import pyplot as plt
# reads an input image
img = cv2.imread('/img1.png',0)
# find frequency of pixels in range 0-255
histr = cv2.calcHist([img],[0],None,[256],[0,256])
# show the plotting graph of an image
plt.plot(histr)
plt.show()
#/*****************************************/


# importing required libraries of opencv
import cv2
# importing library for plotting
from matplotlib import pyplot as plt
# reads an input image
img = cv2.imread('/img2.png',0)
# find frequency of pixels in range 0-255
histr = cv2.calcHist([img],[0],None,[256],[0,256])
```

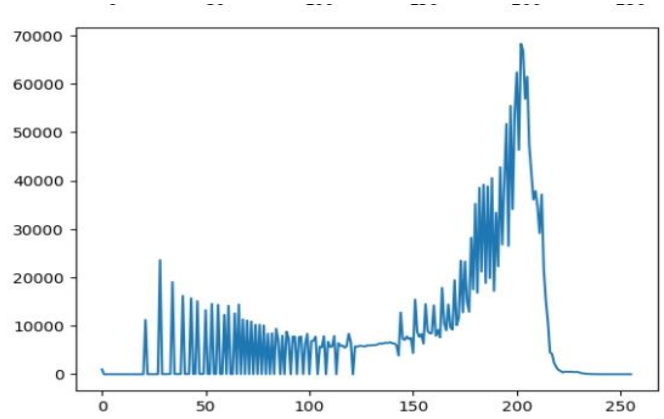
```
# show the plotting graph of an image
plt.plot(histr)
plt.show()
```

Input :

Output :


160000
140000
120000
100000
80000
60000
40000
20000
0
0
50
100
150
200
250


400000
350000
300000
250000
200000
150000
100000
50000
0
0
50
100
150
200
250



2) Aim : Program to apply histogram equalization

Histogram equalization is a method in image processing of contrast adjustment using the image's histogram.This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values. The method is useful in images with backgrounds and foregrounds that are both bright or both dark.

Code :

```
# import Opencv

import cv2



# import Numpy

import numpy as np
```

```
# read a image using imread
img = cv2.imread(\'F:\\do_nawab.png\', 0)

# creating a Histograms Equalization
# of a image using cv2.equalizeHist()
equ = cv2.equalizeHist(img)

# stacking images side-by-side
res = np.hstack((img, equ))

# show image input vs output
cv2.imshow(\'image\', res)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input :

Output :

Practical No 2C

Aim : Write program to apply smoothing and sharpening filters on grayscale and color images low pass and high pass

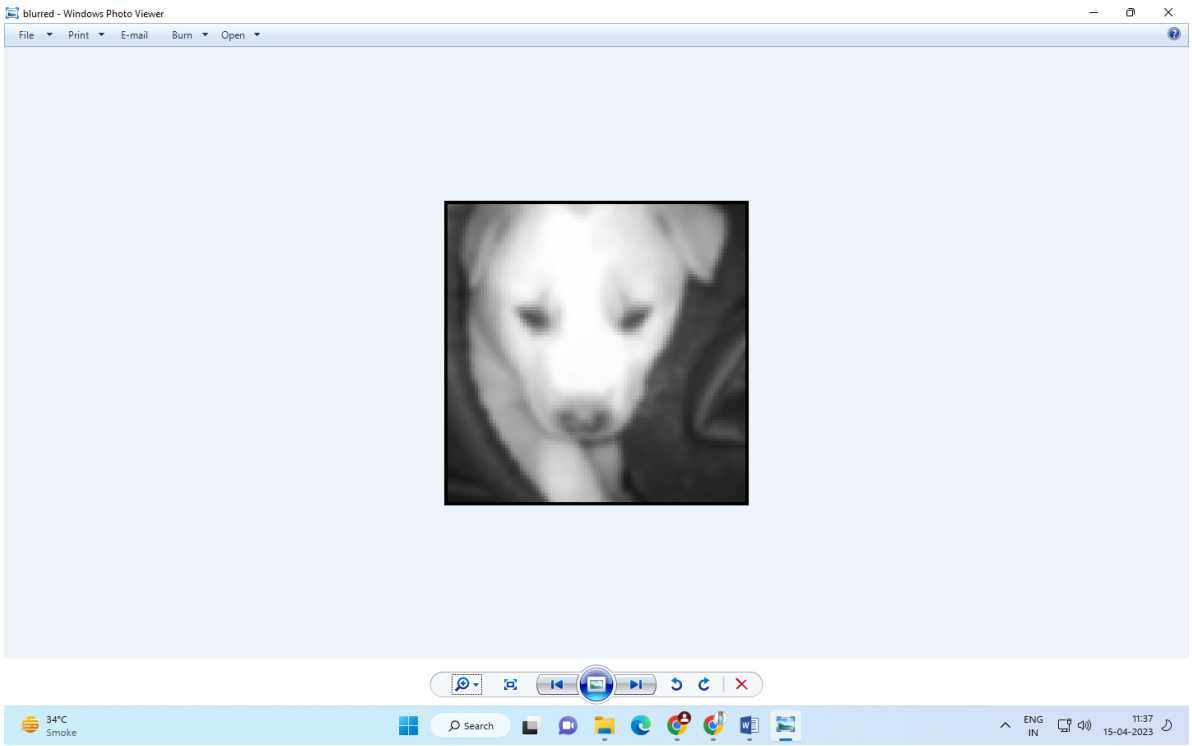
- **Low Pass filtering:** It is also known as the smoothing filter. It removes the high-frequency content from the image. It is also used to blur an image. A low pass averaging filter mask is as shown.

Code :

```
import cv2
import numpy as np


# Read the image
img = cv2.imread('/content/dog.png', 0)

# Obtain number of rows and columns
# of the image
m, n = img.shape

# Develop Averaging filter(3, 3) mask
mask = np.ones([3, 3], dtype = int)
mask = mask / 9

# Convolve the 3X3 mask over the image
img_new = np.zeros([m, n])

for i in range(1, m-1):
    for j in range(1, n-1):
        temp = img[i-1, j-1]*mask[0, 0]+img[i-1, j]*mask[0, 1]+img[i-
1, j + 1]*mask[0, 2]+img[i, j-
1]*mask[1, 0]+ img[i, j]*mask[1, 1]+img[i, j + 1]*mask[1, 2]+img[i + 1, j-
1]*mask[2, 0]+img[i + 1, j]*mask[2, 1]+img[i + 1, j + 1]*mask[2, 2]

        img_new[i, j]= temp

img_new = img_new.astype(np.uint8)
cv2.imwrite('blurred.tif', img_new)
```

Input :

Output :



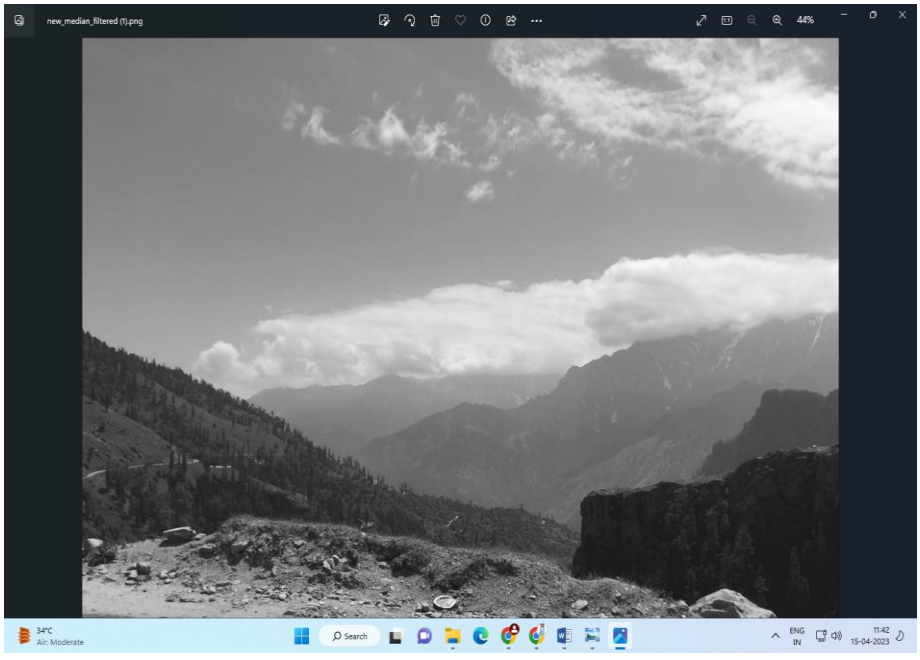
- **High Pass Filtering:** It eliminates low-frequency regions while retaining or enhancing the high-frequency components. A high pass filtering mask is as shown.

  Code :

```
import cv2
import numpy as np


# Read the image
img_noisy1 = cv2.imread('/content/exe.png', 0)

# Obtain the number of rows and columns
# of the image
m, n = img_noisy1.shape

# Traverse the image. For every 3X3 area,
# find the median of the pixels and
# replace the center pixel by the median
img_new1 = np.zeros([m, n])

for i in range(1, m-1):
  for j in range(1, n-1):
    temp = [img_noisy1[i-1, j-1],
      img_noisy1[i-1, j],
      img_noisy1[i-1, j + 1],
      img_noisy1[i, j-1],
      img_noisy1[i, j],
      img_noisy1[i, j + 1],
      img_noisy1[i + 1, j-1],
      img_noisy1[i + 1, j],
      img_noisy1[i + 1, j + 1]]

    temp = sorted(temp)
    img_new1[i, j]= temp[4]

img_new1 = img_new1.astype(np.uint8)
cv2.imwrite('new_median_filtered.png', img_new1)
```

Input :

Output :



Practical No. 2(D)

Aim : Write a program to apply smoothing(Low Pass) and sharpening(High Pass) filters on grayscale and color images

Smoothing filters are also known as low-pass filters. They work by averaging the pixel values in a local neighborhood to reduce high-frequency noise in the image. This results in a blurred image that can be useful for tasks such as denoising or blurring out certain details in an image. One common smoothing filter is the Gaussian filter, which is often used because it preserves edges in an image while smoothing out the noise.

On the other hand, sharpening filters are also known as high-pass filters. They work by enhancing the edges in an image by amplifying the high-frequency components in the image. This results in an image that appears sharper and more detailed. One common sharpening filter is the Laplacian filter, which is often used because it highlights edges in an image.

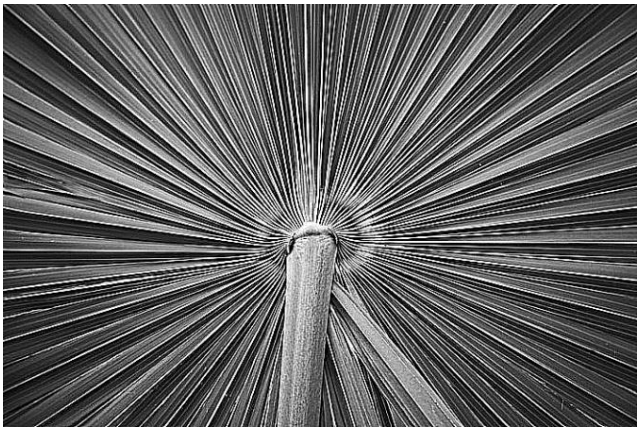
Code:

```
import cv2
import numpy as np

# Load the grayscale and color images
img_gray = cv2.imread('gray_image.jpg', cv2.IMREAD_GRAYSCALE)
img_color = cv2.imread('color_image.jpg', cv2.IMREAD_COLOR)

# Define the smoothing (low-pass) filter
kernel_smooth = np.ones((5,5),np.float32)/25

# Apply the smoothing filter to the grayscale and color images
img_gray_smooth = cv2.filter2D(img_gray,-1,kernel_smooth)
img_color_smooth = cv2.filter2D(img_color,-1,kernel_smooth)

# Define the sharpening (high-pass) filter
kernel_sharpen = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])

# Apply the sharpening filter to the grayscale and color images
img_gray_sharpen = cv2.filter2D(img_gray,-1,kernel_sharpen)
img_color_sharpen = cv2.filter2D(img_color,-1,kernel_sharpen)

# Display the grayscale images with the applied filters
#cv2.imshow('Original Grayscale', img_gray)
cv2.imshow('Smoothed Grayscale', img_gray_smooth)
```

```
cv2.imshow('Sharpened Grayscale', img_gray_sharpen)


# Display the grayscale images with the applied filters
#cv2.imshow('Original Color', img_color)
cv2.imshow('Smoothed Color', img_color_smooth)
cv2.imshow('Sharpened Color', img_color_sharpen)

# Wait for a key press and then close all windows
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

Output:



Practical No. 3(A)

Aim : Program to apply Discrete Fourier Transform on an image

The Discrete Fourier Transform (DFT) is a mathematical technique used to transform a discrete sequence of data from the time domain into the frequency domain. It is widely used in signal processing, image processing, audio analysis, and many other fields.
The DFT converts a sequence of N complex-valued samples x(n), for n=0,1, ..., N-1, into a sequence of N complex-valued samples X(k), for k=0,1, ..., N-1,

where: X(k) = sum(x(n) * exp(-j2pikn/N)), for n=0,1, ..., N-1

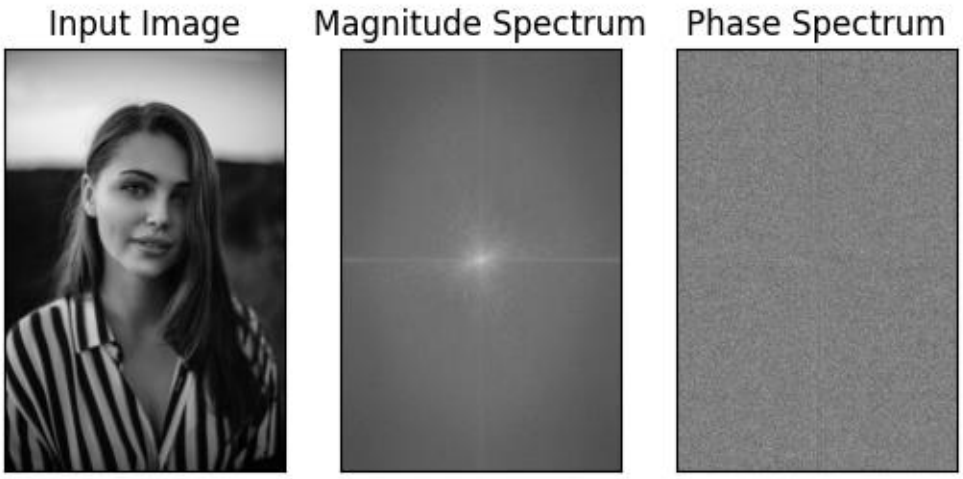
Here, j is the imaginary unit (sqrt (-1)), and pi is the mathematical constant pi. The term exp(-j2pikn/N) is called the twiddle factor, and represents a complex sinusoidal wave of frequency k/N in the discrete Fourier transform.

Code:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load the input image in grayscale mode
img = cv2.imread('/content/image.jpg', 0)

# Compute the Discrete Fourier Transform of the input image
img_dft = np.fft.fft2(img)
img_dft_shift = np.fft.fftshift(img_dft)

# Compute the magnitude and phase spectra of the DFT
magnitude_spectrum = 20 * np.log(np.abs(img_dft_shift))
phase_spectrum = np.angle(img_dft_shift)

# Display the original image and its DFT magnitude and phase spectra
plt.subplot(1, 3, 1), plt.imshow(img, cmap='gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])

plt.subplot(1, 3, 2), plt.imshow(magnitude_spectrum, cmap='gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])

plt.subplot(1, 3, 3), plt.imshow(phase_spectrum, cmap='gray')
plt.title('Phase Spectrum'), plt.xticks([]), plt.yticks([])

plt.show()
```

Input:

Output:



Practical No. 3(B)

Aim : Program to apply Low Pass & High Pass filters in Frequency Domain

Low-pass and high-pass filters are common types of filters used in image processing. Low-pass filters allow low-frequency components to pass through while attenuating high-frequency components. High-pass filters, on the other hand, allow high-frequency components to pass through while attenuating low-frequency components.

Code:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load the input image in grayscale mode
img = cv2.imread('image.jpg', 0)

# Compute the Fast Fourier Transform of the input image
img_fft = np.fft.fft2(img)
img_fft_shift = np.fft.fftshift(img_fft)

# Define the size of the low pass filter and create the filter mask
rows, cols = img.shape
crow, ccol = rows // 2, cols // 2
low_pass_size = 30
mask_low_pass = np.zeros((rows, cols), np.uint8)
mask_low_pass[crow - low_pass_size:crow + low_pass_size, ccol -
low_pass_size:ccol + low_pass_size] = 1

# Apply the low pass filter mask in frequency domain
img_fft_shift_low_pass = img_fft_shift * mask_low_pass

# Compute the Inverse Fast Fourier Transform of the filtered image
img_low_pass = np.fft.ifft2(np.fft.ifftshift(img_fft_shift_low_pass))
img_low_pass = np.abs(img_low_pass)

# Define the size of the high pass filter and create the filter mask
high_pass_size = 30
mask_high_pass = np.ones((rows, cols), np.uint8)
mask_high_pass[crow - high_pass_size:crow + high_pass_size, ccol -
high_pass_size:ccol + high_pass_size] = 0

# Apply the high pass filter mask in frequency domain
img_fft_shift_high_pass = img_fft_shift * mask_high_pass
```

```
# Compute the Inverse Fast Fourier Transform of the filtered image
img_high_pass = np.fft.ifft2(np.fft.ifftshift(img_fft_shift_high_pass))
img_high_pass = np.abs(img_high_pass)

# Display the original and filtered images
plt.subplot(2, 2, 1), plt.imshow(img, cmap='gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])

plt.subplot(2, 2, 2), plt.imshow(np.log(1 + np.abs(img_fft_shift)), cmap='gray')
plt.title('Frequency Spectrum'), plt.xticks([]), plt.yticks([])

plt.subplot(2, 2, 3), plt.imshow(img_low_pass, cmap='gray')
plt.title('Low Pass Filtered Image'), plt.xticks([]), plt.yticks([])

plt.subplot(2, 2, 4), plt.imshow(img_high_pass, cmap='gray')
plt.title('High Pass Filtered Image'), plt.xticks([]), plt.yticks([])

plt.show()
```

Input:

Output:

Original Image

Frequency Spectrum

Low Pass Filtered Image

High Pass Filtered Image

Practical No. 3(C)

Aim: Program to apply laplacian filter in frequency domain

The Laplacian filter in the frequency domain, we can use the fact that the Laplacian operator in the spatial domain is equivalent to multiplying the Fourier transform of an image by the Laplacian filter kernel in the frequency domain.

The Laplacian filter kernel in the frequency domain is given by:

H(u,v) = -4 * pi^2 * (u^2 + v^2)

where u and v are the spatial frequencies in the horizontal and vertical directions, respectively, and pi is the mathematical constant pi.

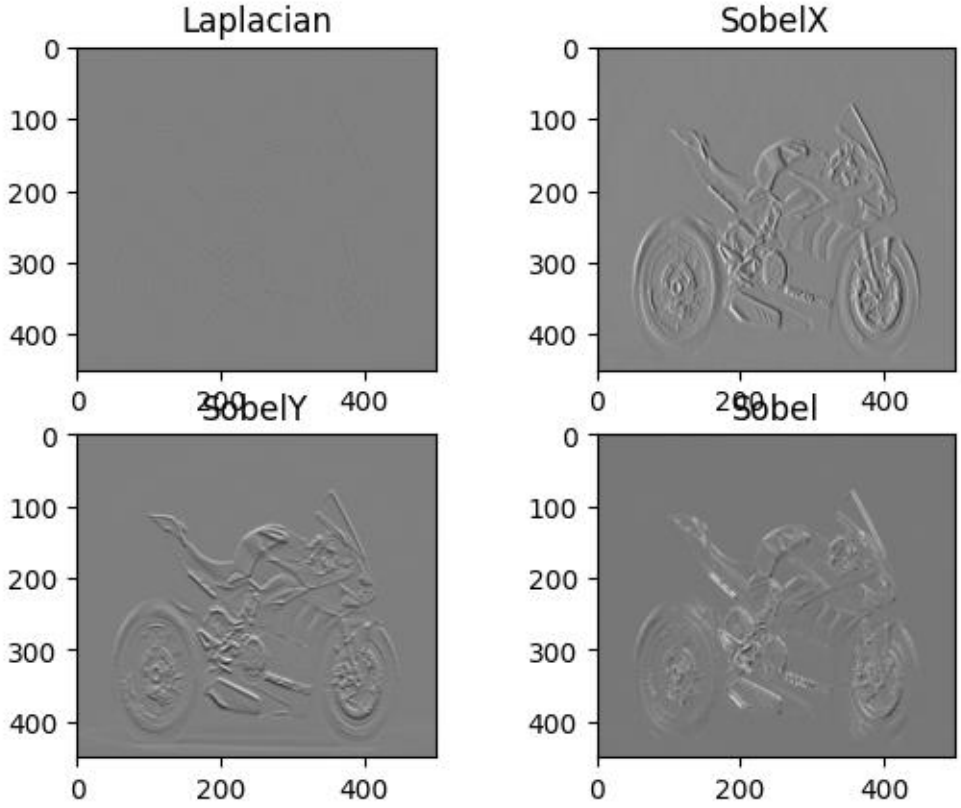
Code:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# read image
img = cv2.imread("/content/Geeks.jpeg")

# resize image
img = cv2.resize(img, (500, 450), interpolation=cv2.INTER_CUBIC)

# convert image to gray scale image
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# apply laplacian blur
laplacian = cv2.Laplacian(gray, cv2.CV_64F)

# sobel x filter where dx=1 and dy=0
sobelx = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=7)

# sobel y filter where dx=0 and dy=1
sobely = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=7)

# combine sobel x and y
sobel = cv2.bitwise_and(sobelx, sobely)
```

```
# plot images
plt.subplot(2, 2, 1)
plt.imshow(laplacian, cmap='gray')
plt.title('Laplacian')

plt.subplot(2, 2, 2)
plt.imshow(sobelx, cmap='gray')
plt.title('SobelX')

plt.subplot(2, 2, 3)
plt.imshow(sobely, cmap='gray')
plt.title('SobelY')

plt.subplot(2, 2, 4)
plt.imshow(sobel, cmap='gray')
plt.title('Sobel')

plt.show()
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

Output:



Practical No. 3(D)

Aim : All other filters can be applied, studied and compared with filters in spatial domain

1) Median Filter (Spatial Domain):

The median filter is a spatial domain filter that is commonly used for removing salt-and-pepper noise from images. It replaces each pixel's value with the median value of its neighboring pixels. This filter helps to smooth the image while preserving edges and fine details.

2) Gaussian Filter (Spatial Domain):

The Gaussian filter is a spatial domain filter that applies a Gaussian distribution to blur an image. It convolves the image with a Gaussian kernel, which attenuates high-frequency components and preserves low-frequency components. This filter is useful for reducing noise and removing high-frequency details.

3) Band-stop Filter (Frequency Domain):

The band-stop filter is a frequency domain filter that attenuates a specific range of frequencies while allowing other frequencies to pass through. In the program, a circular mask with a hole is created to remove a band of frequencies. This filter is effective in removing specific periodic noise or interference from an image.

4) Adaptive Filter (Frequency Domain):

The adaptive filter is a frequency domain filter that selectively keeps frequency components based on an adaptive threshold. In the program, frequencies with magnitudes above the threshold are retained, while those below are set to zero. This filter is useful for preserving important frequency information while reducing noise or unwanted artifacts.

Code:

```
import cv2
import numpy as np
from scipy import fftpack

# Load the image
image = cv2.imread('input_image.jpg', 0)
```

```
# Spatial Domain Filters
# 1. Median Filter
median_filtered = cv2.medianBlur(image, 5)

# 2. Gaussian Filter
gaussian_filtered = cv2.GaussianBlur(image, (5, 5), 0)

# Frequency Domain Filters
# Perform 2D Fast Fourier Transform (FFT)
fft_image = fftpack.fftshift(fftpack.fft2(image))

# 1. Band-stop Filter in Frequency Domain
rows, cols = image.shape
crow, ccol = rows // 2, cols // 2
band_stop_radius = 30
band_stop_width = 10
mask_band_stop = np.ones((rows, cols), np.uint8)
mask_band_stop[crow - band_stop_radius:crow + band_stop_radius,
               ccol - band_stop_width//2:ccol + band_stop_width//2] = 0
mask_band_stop[crow - band_stop_width//2:crow + band_stop_width//2,
               ccol - band_stop_radius:ccol + band_stop_radius] = 0
fft_image_band_stop = fft_image * mask_band_stop
inverse_band_stop = np.abs(fftpack.ifft2(fftpack.ifftshift(fft_image_band_stop)))

# 2. Adaptive Filter in Frequency Domain
adaptive_threshold = 50
fft_image_adaptive = np.where(np.abs(fft_image) > adaptive_threshold, fft_image, 0)
inverse_adaptive = np.abs(fftpack.ifft2(fftpack.ifftshift(fft_image_adaptive)))
```

```
# Display the results
cv2.imshow('Median Filter (Spatial Domain)', median_filtered)
cv2.imshow('Gaussian Filter (Spatial Domain)', gaussian_filtered)
cv2.imshow('Band-stop Filter (Frequency Domain)', inverse_band_stop)
cv2.imshow('Adaptive Filter (Frequency Domain)', inverse_adaptive)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

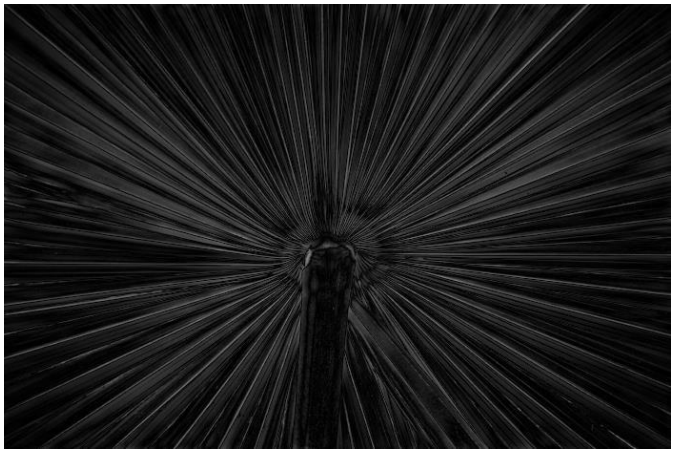
Input:

Output:

Median Filter (Spatial Domain)

Gaussian Filter (Spatial Domain)

Band-stop Filter (Frequency Domain)



Adaptive Filter (Frequency Domain)

Practical No. 3(E)

Aim : Program to perform High frequency emphasis filtering, High boost filtering and Homomorphic filtering

Input Image:

1) High-frequency emphasis filtering: This technique enhances the high-frequency components of an image, which correspond to the edges and fine details. It does so by applying a high-pass filter to the image and adding the result back to the original image with a specified weight. This enhances the edges and fine details while preserving the overall image brightness and contrast.

   Code:

```
import cv2
import numpy as np

def high_frequency_emphasis_filter(img, alpha=1.5, beta=0.5):
    # Convert image to grayscale
    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Apply Laplacian filter to obtain the high-frequency components
    laplacian_img = cv2.Laplacian(gray_img, cv2.CV_32F)

    # Apply high-frequency emphasis filter
    filtered_img = np.uint8(alpha * laplacian_img + beta * gray_img)

    return filtered_img
```

```
# Load image
img = cv2.imread("image.jpg")

# Apply high-frequency emphasis filter
filtered_img = high_frequency_emphasis_filter(img)

# Display filtered image
cv2.imshow("Filtered Image", filtered_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:

2) High boost filtering: This is a special case of high-frequency emphasis filtering, where the weight added to the high-pass filtered image is higher than 1. This results in a sharper and more contrasted image, but also introduces the risk of amplifying noise and artifacts.

Code:

```
import cv2
import numpy as np

# Load image from file
img = cv2.imread('input_image.jpg')

# Convert image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Create filter kernel for high-pass filter
ksize = 5
kernel = np.zeros((ksize, ksize), np.float32)
```

```
kernel[ksize//2, ksize//2] = 2.0
kernel = cv2.GaussianBlur(kernel, (ksize, ksize), 0)
kernel /= np.sum(kernel)

# Apply high-pass filter to grayscale image
filtered = cv2.filter2D(gray, -1, kernel)

# Define the boost factor
boost_factor = 1.5

# Compute the high-boost filtered image
high_boost = gray + boost_factor * filtered

# Clip the image to the range [0, 255]
high_boost = np.clip(high_boost, 0, 255)

# Convert high-boost filtered image to RGB
high_boost_rgb = cv2.cvtColor(high_boost.astype(np.uint8),
cv2.COLOR_GRAY2BGR)

# Display high-boost filtered image
cv2.imshow('High-Boost Filtered Image', high_boost_rgb)
cv2.waitKey(0)

# Save high-boost filtered image to file
cv2.imwrite('output_image.jpg', high_boost_rgb)
```

Output:

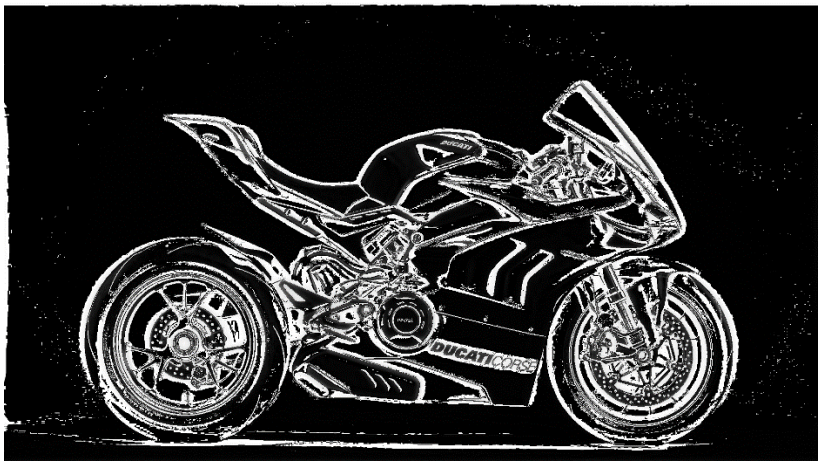
3) Homomorphic filtering: This technique is used for removing the effects of uneven illumination in an image, which can make it difficult to distinguish the details and textures. It does so by separating the image into its low-frequency and high-frequency components, and then applying different filters to each component before recombining them. This results in an image with improved contrast and clarity.

Code:

```
import numpy as np
import cv2
from scipy.signal import fftconvolve

def homomorphic_filter(img, gamma_l=0.2, gamma_h=2.5, c=1, d0=50):
    # Convert image to log domain
    img = np.log1p(np.array(img, dtype="float") / 255)

    # Create Gaussian high-pass filter
    rows, cols = img.shape
    x = np.linspace(-0.5, 0.5, cols) * cols
    y = np.linspace(-0.5, 0.5, rows) * rows
    xx, yy = np.meshgrid(x, y)
    filter = (gamma_h - gamma_l) * (1 - np.exp(-c * (xx**2 + yy**2) / d0**2)) +
gamma_l

    # Apply filter to image
    filtered_img = np.real(np.fft.ifft2(np.fft.fft2(img) * np.fft.fftshift(filter)))

    # Convert back to linear domain
    filtered_img = np.exp(filtered_img) - 1

    # Scale to 0-255 and convert to uint8
    filtered_img = np.uint8(filtered_img / np.max(filtered_img) * 255)

    return filtered_img

# Load image
img = cv2.imread("image.jpg", 0)

# Apply homomorphic filter
filtered_img = homomorphic_filter(img)

# Display filtered image
cv2.imshow("Filtered Image", filtered_img)
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:



Practical No. 4(A)

Aim : Program to denoise using spatial mean, median and adaptive filtering

Spatial Mean Filtering:
The spatial mean filter is a simple technique that replaces each pixel's value with the mean value of its neighboring pixels. It helps reduce noise by averaging out the pixel intensities.

Spatial Median Filtering:
The spatial median filter is another popular technique that replaces each pixel's value with the median value of its neighboring pixels. It is particularly effective in preserving edges while reducing noise.

Adaptive Mean Filtering:
The adaptive mean filter is a local filtering technique that applies a thresholding operation based on the mean value of the pixel's neighborhood. It helps reduce noise while preserving local image details.

Code:

```
import cv2
import numpy as np

# Load the noisy image
image = cv2.imread('noisy_image.jpg', cv2.IMREAD_GRAYSCALE)

# Spatial Mean Filtering
mean_filtered = cv2.blur(image, (3, 3))

# Spatial Median Filtering
median_filtered = cv2.medianBlur(image, 3)

# Adaptive Mean Filtering
adaptive_filtered = cv2.adaptiveThreshold(image, 255,
cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 5, 2)

# Display the original and denoised images
cv2.imshow('Spatial Mean Filtered Image', mean_filtered)
cv2.imshow('Spatial Median Filtered Image', median_filtered)
cv2.imshow('Adaptive Mean Filtered Image', adaptive_filtered)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

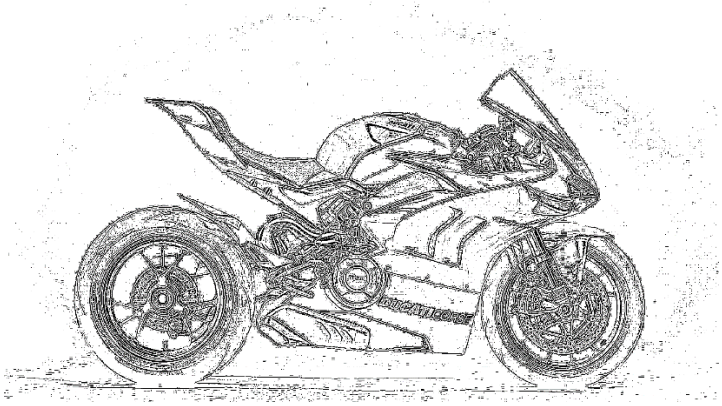
Input:

Output:

Spatial Mean Filtered Image

Spatial Median Filtered Image

Adaptive Mean Filtered Image



Practical No. 4(B)

Aim : Program to Image Deblurring using Inverse, Wiener Filter

Image deblurring is a common problem in image processing and computer vision, where the goal is to recover the original sharp image from a blurred or degraded version of the image. Blurring can occur due to various factors such as camera shake, motion blur, defocus blur, etc. In this context, two popular techniques for image deblurring are inverse filtering and Wiener filtering.

Inverse filtering is a simple technique that involves dividing the Fourier transform of the blurred image by the Fourier transform of the blurring kernel. The resulting quotient is then inverse transformed back to the spatial domain to obtain the deblurred image. However, this method is highly sensitive to noise and can produce ringing artifacts in the image.

To overcome these issues, the Wiener filter was introduced as a more robust deblurring technique. The Wiener filter is a frequency-dependent filter that aims to minimize the mean square error between the original image and the blurred image, subject to some constraints on the filter parameters. The Wiener filter considers the power spectral density of the blurred image and the noise level to compute a frequency-dependent gain factor that is applied to the Fourier transform of the blurred image. The result is then inverse transformed to obtain the deblurred image.

Code:

```
import cv2

import numpy as np

# Load blurred image

img_blur = cv2.imread('blurred_image.jpg', cv2.IMREAD_GRAYSCALE)

# Apply inverse filter

H_inv = np.fft.fft2(img_blur)

H_inv = np.where(H_inv != 0, 1 / H_inv, 0)

img_inv = np.fft.ifft2(H_inv * np.fft.fft2(img_blur)).real

# Apply Wiener filter

H_wiener = np.fft.fft2(img_blur)

k = 0.1  # regularization parameter

H_wiener = np.where(np.abs(H_wiener) ** 2 >= k, np.conj(H_wiener) /
(np.abs(H_wiener) ** 2 + k), 0)
```

```
img_wiener = np.fft.ifft2(H_wiener * np.fft.fft2(img_blur)).real
# Display results
cv2.imshow('Blurred Image', img_blur)
cv2.imshow('Inverse Filter', img_inv.astype(np.uint8))
cv2.imshow('Wiener Filter', img_wiener.astype(np.uint8))
cv2.waitKey(0)
# Save results
cv2.imwrite('inverse_filtered_image.jpg', img_inv)
cv2.imwrite('wiener_filtered_image.jpg', img_wiener)
```

Input:

Output:

Blurred Image

Inverse Filter

Wiener Filter

Practical No.5(A)

Aim : Program to read a color image and segment into RGB planes , histogram of color image

RGB image can be viewed as three different images(a red scale image, a green scale image and a blue scale image) stacked on top of each other, and when fed into the red, green and blue inputs of a color monitor, it produces a color image on the screen.

- RGB color model is the model in which Red, Blue, and Green colors are blended together to form an array of colors
- In this article, we will learn the concept of extraction of RGB components from an image and the calculation of RGB values pixels on the MATLAB interface.
- An RGB image is sometimes referred to as a true color image as the precision with which a real-life image can be replicated has led to the nickname “true color image.”

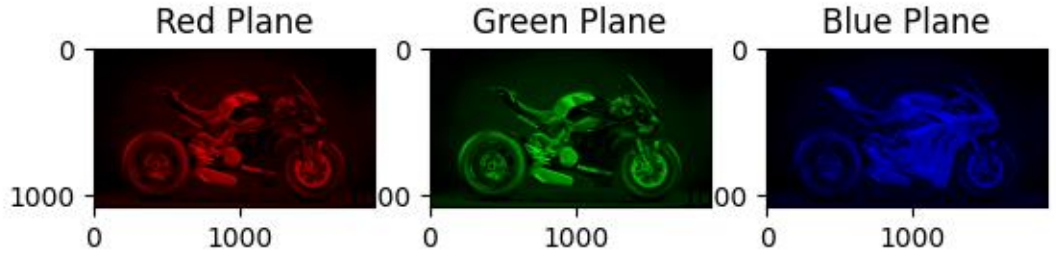
Code:

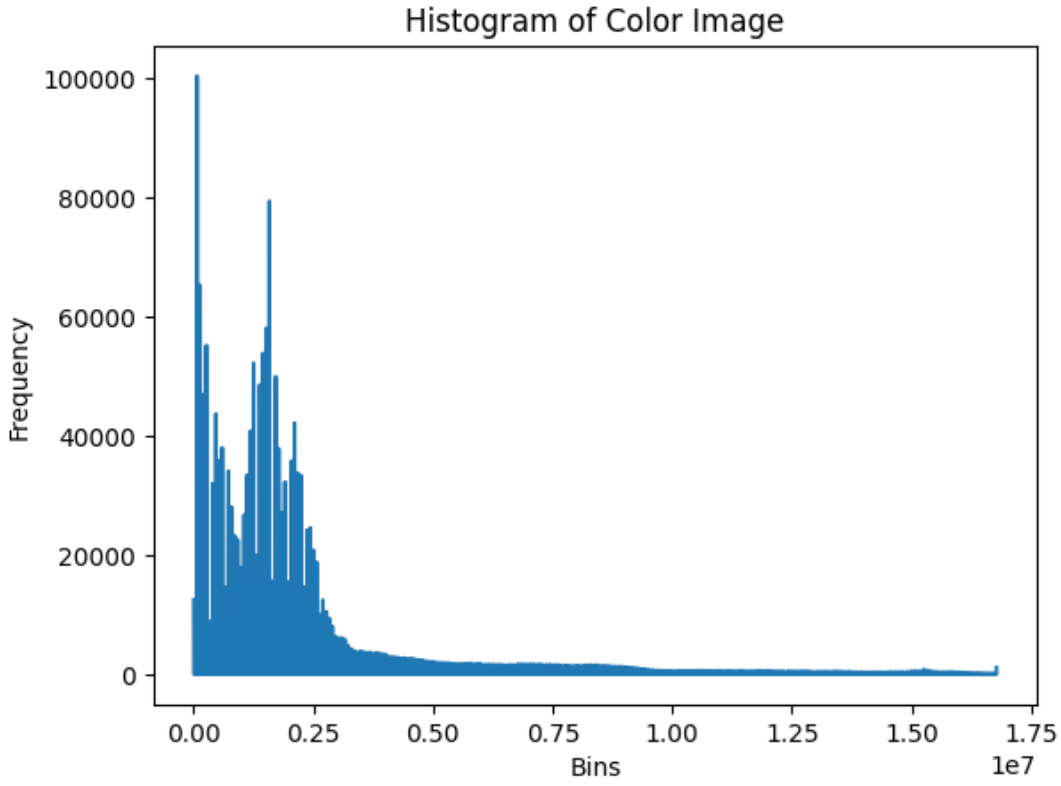
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Read the color image
image = cv2.imread('/content/ducati.jpg')
# Rows and columns in the image
r, c = image.shape[:2]

# Creating zero matrices
R = np.zeros((r, c, 3), dtype=np.uint8)
G = np.zeros((r, c, 3), dtype=np.uint8)
B = np.zeros((r, c, 3), dtype=np.uint8)

# Storing the corresponding color plane

# Red plane
R[:, :, 0] = image[:, :, 0]

# Green plane
G[:, :, 1] = image[:, :, 1]

# Blue plane
B[:, :, 2] = image[:, :, 2]

# Displaying the images
```

```
plt.subplot(1, 3, 1)
plt.title('Red Plane')
plt.imshow(R)

plt.subplot(1, 3, 2)
plt.title('Green Plane')
plt.imshow(G)

plt.subplot(1, 3, 3)
plt.title('Blue Plane')
plt.imshow(B)

plt.show()

# Calculate the histogram of the color image
histogram = cv2.calcHist([image], [0, 1, 2], None, [256, 256, 256], [0, 256, 0, 256, 0, 256])

# Flatten the histogram array
histogram_flat = histogram.flatten()

# Plot the histogram
plt.figure()
plt.title('Histogram of Color Image')
plt.xlabel('Bins')
plt.ylabel('Frequency')
plt.plot(histogram_flat)
plt.show()
```

Input:

Output:


Red Plane
Green Plane
Blue Plane
0
1000
0
1000


Histogram of Color Image
Frequency
100000
80000
60000
40000
20000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Bins
1e7

Practical No. 5(B)

Aim : Program for converting from one color model to another model

An RGB (colored) image has three channels, Red, Blue and Green. A colored image in OpenCV has a shape in [H, W, C] format, where H, W, and C are image height, width and number of channels. All three channels have a value range between 0 and 255.

The HSV image also has three channels, the Hue, Saturation and Value channels. In OpenCV, the values of the Hue channel range from 0 to 179, whereas the Saturation and Value channels range from 0 to 255.

In OpenCV, to convert an RGB image to HSV image, we use the cv2.cvtColor() function. This function is used to convert an image from one color space to another

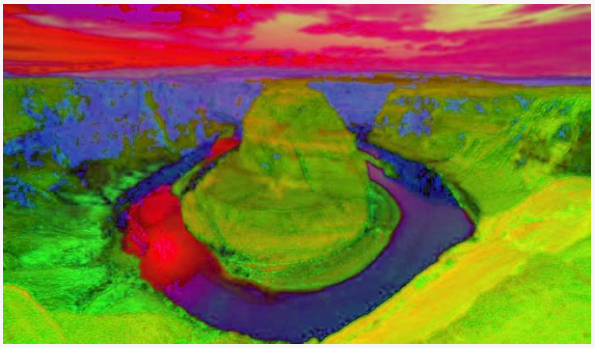
Code:

```
import cv2
```

```
from google.colab.patches import cv2_imshow

# read the input RGB image as BGR format
bgr_img = cv2.imread('/content/water.jpg')

# Convert the BGR image to HSV Image
hsv_img = cv2.cvtColor(bgr_img, cv2.COLOR_BGR2HSV)
cv2.imwrite('hsv_image.jpg', hsv_img)

# Display the HSV image
cv2_imshow(hsv_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

Output:



Practical No. 5(C)

Aim : Program to apply false colouring(pseudo) on a gray scale image

Pseudocolor can be a useful tool for enhancing the contrast and visualizing your data more easily. This is especially useful when making presentations of your data using projectors(because their contrast is typically quite poor). Pseudocolor is only relevant to single-channel, grayscale, luminosity images

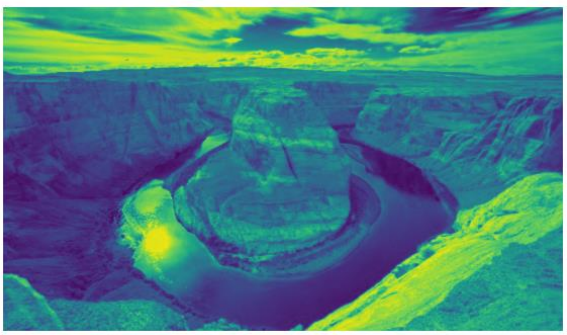
Code:

```
from matplotlib import pyplot as plt, image as mimg
```

```
plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True

img = mimg.imread('/content/water.jpg')
lum_img = img[:, :, 0]

plt.imshow(lum_img)
plt.axis('off')

plt.show()
```

Input:

Output:



Practical No. 6

Aim : Program to compute Discrete Cosine Transforms, Walsh -Hadamard Transforms, Haar Transform , Wavelet

Input:

1) Discrete Cosine Transform (DCT):
The Discrete Cosine Transform (DCT) is a popular transform used in image processing. It converts a spatial domain image into the frequency domain, similar to the Fourier Transform. However, unlike the Fourier Transform, the DCT only uses real numbers and is particularly effective for compactly representing images with smooth variations. The DCT coefficients represent the image's frequency components, with lower frequencies concentrated in the lower frequency coefficients. In image compression, the DCT is widely used, such as in the JPEG image compression algorithm.

Code:

```
import cv2
import numpy as np
from scipy.fftpack import dct, fft, ifft
from scipy.linalg import hadamard
from google.colab.patches import cv2_imshow

# Load the image
image = cv2.imread('/content/leaf.jpg', 0)

# Discrete Cosine Transform (DCT)
dct_image = dct(image.astype(float), norm='ortho')
inverse_dct = dct(dct_image, type=3, norm='ortho')

cv2_imshow(inverse_dct.astype(np.uint8))
```

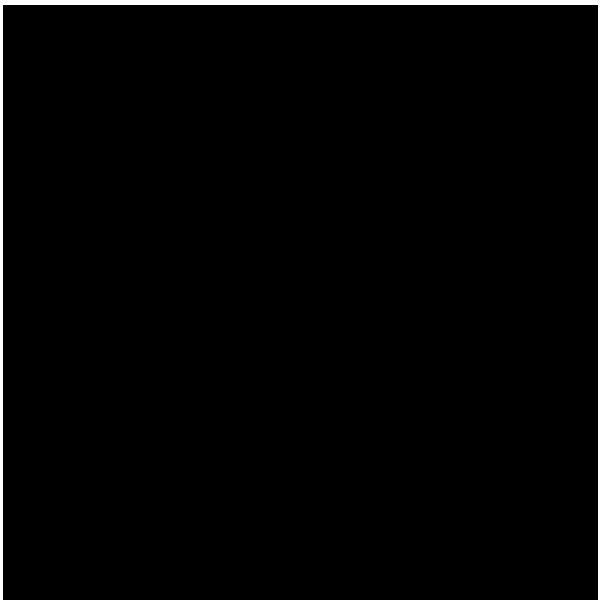
Output:

2) Walsh-Hadamard Transform (WHT):
The Walsh-Hadamard Transform (WHT) is another commonly used transform in image processing. It is based on the Hadamard matrix, which is a square matrix with entries of +1 and -1. The WHT converts an image into a set of Walsh-Hadamard coefficients, representing different frequency components. The WHT is particularly useful for pattern recognition, data compression, and image watermarking. It has applications in image and video compression, where it can efficiently represent spatial information.

Code:

```
import cv2
import numpy as np
from scipy.fftpack import dct, fft, ifft
from scipy.linalg import hadamard

# Load the image
image = cv2.imread('input_image.jpg', 0)

# Resize or crop the image to a power of 2
resized_image = cv2.resize(image, (power_of_2, power_of_2))

# Walsh-Hadamard Transform (WHT)
wht_matrix = hadamard(resized_image.shape[0])
wht_image = wht_matrix @ resized_image @ wht_matrix
inverse_wht = wht_matrix @ wht_image @ wht_matrix

cv2.imshow('Walsh-Hadamard Transform', inverse_wht.astype(np.uint8))
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:



3) Haar Transform:
The Haar Transform is a specific type of wavelet transform used in image processing. It is a fast and computationally efficient transform that separates an image into two components: approximation and detail. The Haar Transform uses a Haar wavelet, which consists of a simple pair of basis functions representing the image's horizontal and vertical edges. It captures both spatial and frequency information of an image and is widely used in image and signal processing applications.

Code:

```
import numpy as np
from scipy.fftpack import dct, fft, ifft
from scipy.linalg import hadamard

# Load the image
image = cv2.imread('input_image.jpg', 0)

# Haar Transform
haar_image = cv2.dct(resized_image.astype(float))
inverse_haar = cv2.idct(haar_image)

cv2.imshow('Haar Transform', inverse_haar.astype(np.uint8))
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:

4) Wavelet Transform:

The Wavelet Transform is a versatile transform used in image processing for analyzing and representing images in both the spatial and frequency domains. Unlike the Fourier Transform that uses sinusoidal functions, wavelet transforms use wavelets, which are small, localized, and time-limited functions. The Wavelet Transform allows analysis of images at different scales and resolutions, making it well-suited for handling images with both localized and global features. It provides a multi-resolution decomposition of an image, separating it into approximation and detail coefficients. Wavelet transforms find applications in image denoising, edge detection, compression, and feature extraction.

Code:

```
import cv2
import numpy as np
from scipy.fftpack import dct, fft, ifft
from scipy.linalg import hadamard

# Load the image
image = cv2.imread('input_image.jpg', 0)

# Resize or crop the image to a power of 2
resized_image = cv2.resize(image, (power_of_2, power_of_2))

# Wavelet Transform
wavelet_image = fft.ifft2(fft.fftshift(fft.fft2(resized_image)))
inverse_wavelet = np.real(fft.ifft2(fft.fftshift(wavelet_image)))

# Display the results
cv2.imshow('Wavelet Transform', inverse_wavelet.astype(np.uint8))
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:

Practical no:8A

Aim: Program to apply erosion, dilation, opening closing

Morphological operations are a set of operations that process images based on shapes. They apply a structuring element to an input image and generate an output image.
The most basic morphological operations are two: Erosion and Dilation
Basics of Erosion:

- Erodes away the boundaries of the foreground object
- Used to diminish the features of an image.

- **EROSION AND DILATION**

Code:

```
# Python program to demonstrate erosion and
# dilation of images.
import cv2
from google.colab.patches import cv2_imshow
import numpy as np

# Reading the input image
img = cv2.imread('/content/geeksforgeeks.png', 0)

# Taking a matrix of size 5 as the kernel
kernel = np.ones((5, 5), np.uint8)
# The first parameter is the original image,
# kernel is the matrix with which image is
# convolved and third parameter is the number
# of iterations, which will determine how much
# you want to erode/dilate a given image.
img_erosion = cv2.erode(img, kernel, iterations=1)
img_dilation = cv2.dilate(img, kernel, iterations=1)
cv2_imshow(img)
cv2_imshow(img_erosion)
cv2_imshow(img_dilation)

cv2.waitKey(0)
```

Input image:

Output image:

Erosion

Dilation

- **OPENING PROCESS**

```
# Python program to demonstrate opening process
import cv2
from google.colab.patches import cv2_imshow
import numpy as np

# Reading the input image
img = cv2.imread('/content/geeksforgeeks.png', 0)

# Taking a matrix of size 5 as the kernel
kernel = np.ones((5, 5), np.uint8)

img_erosion = cv2.erode(img, kernel, iterations=1)
img_dilation = cv2.dilate(img_erosion, kernel, iterations=1)

cv2_imshow(img_erosion)
cv2_imshow(img_dilation)

cv2.waitKey(0)
```

Output image:

- **CLOSING PROCESS**

```
# Python program to demonstrate closing process
import cv2
from google.colab.patches import cv2_imshow
import numpy as np

# Reading the input image
img = cv2.imread('/content/geeksforgeeks.png', 0)

# Taking a matrix of size 5 as the kernel
kernel = np.ones((5, 5), np.uint8)

img_dilation = cv2.dilate(img, kernel, iterations=1)
img_erosion = cv2.erode(img_dilation, kernel, iterations=1)

cv2_imshow(img_dilation)
cv2_imshow(img_erosion)
cv2.waitKey(0)
```

Output image

Practical no: 8B

Aim: Program to detecting the boundary of an image

Edge Detection, is an Image Processing discipline that incorporates mathematics methods to find edges in a Digital Image. Edge Detection internally works by running a filter/Kernel over a Digital Image, which detects discontinuities in Image regions like stark changes in brightness/Intensity value of pixels. **There are two forms of edge detection:**

- Search Based Edge detection (First order derivative)
- Zero Crossing Based Edge detection (Second order derivative)

Code:

```
from PIL import Image, ImageFilter

img = Image.open(r"/content/edge detection.png")

# Converting the image to grayscale, as Sobel Operator requires

# input image to be of mode Grayscale (L)

img = img.convert("L")

# Calculating Edges using the passed laplacian Kernel

final = img.filter(ImageFilter.Kernel((3, 3), (-1, -1, -1, -1, 8,-1, -1, -1, -1), 1, 0))

final.save("EDGE_sample.png")
```

Input image:

Output image:

Practical No: 8C

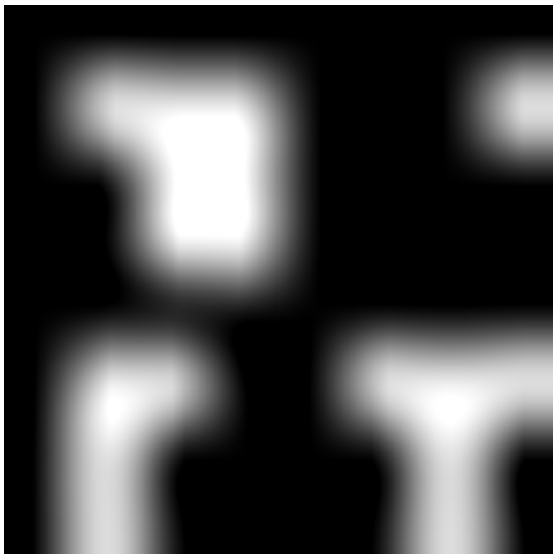
Aim: Program to apply Hit-or-Miss Transform

The hit-and-miss transform is a general binary morphological operation that can be used to look for particular patterns of foreground and background pixels in an image. It is actually the basic operation of binary morphology since almost all the other binary morphological operators can be derived from it. As with other binary morphological operators it takes as input a binary image and a structuring element, and produces another binary image as output.

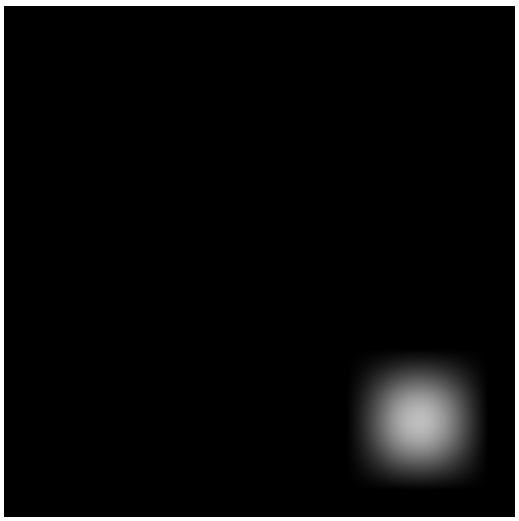
Code:

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Create the input image
input_image = np.array((
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 255, 255, 255, 0, 0, 0, 255],
    [0, 0, 255, 255, 0, 0, 0, 0],
    [0, 0, 255, 255, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 255, 255, 0, 0, 255, 255, 255],
    [0,255, 0, 0, 0, 0, 255, 0],
    [0, 255, 0, 0, 0, 0, 255, 0]), dtype="uint8")

# Construct the structuring element
kernel = np.array((
        [1, 1, 1],
        [0, 1, -1],
        [0, 1, -1]), dtype="int")

# Apply hit-or-miss transformation
output_image = cv2.morphologyEx(input_image, cv2.MORPH_HITMISS, kernel)
cv2_imshow(input_image)
cv2_imshow(output_image)
```

Input Image:



Output Image:



Practical No: 8D

Aim: Program to apply morphological gradient on an image

Morphological operation is elaborated that is **Gradient**. It is used for generating the outline of the image. There are two types of gradients, internal and external gradients. The internal gradient enhances the internal boundaries of objects brighter than their background and external boundaries of objects darker than their background. For binary images, the internal gradient generates a mask of the internal boundaries of the foreground image objects.

Code:

```
import numpy as np
from google.colab.patches import cv2_imshow
import cv2

from matplotlib import pyplot as plt

image = cv2.imread('/content/geeksforgeeks.png',0)
retVal,mask = cv2.threshold(image,155,255,cv2.THRESH_BINARY_INV)
kernel = np.ones((7,7),np.uint8)
gradient = cv2.morphologyEx(mask,cv2.MORPH_GRADIENT,kernel)
cv2_imshow(gradient)
```

Input image:

Output image:

Practical No : 8E

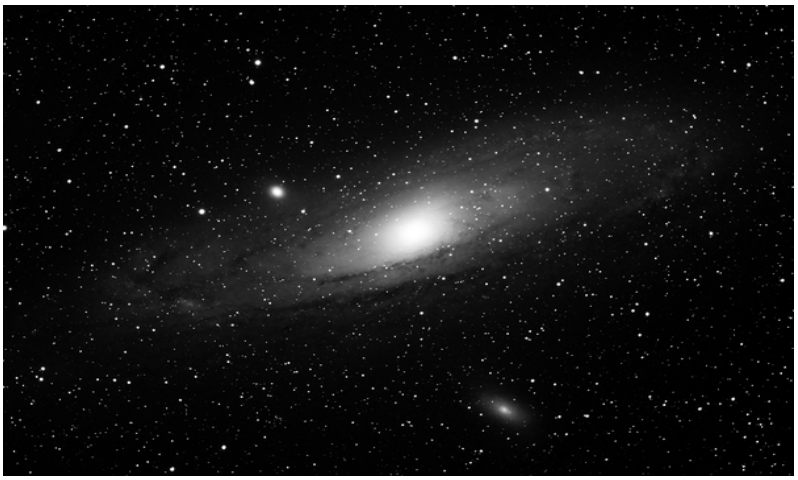
Aim: Program to apply Top-Hat/Bottom-Hat Transformations

In morphology and digital image processing, top-hat and black-hat transform are operations that are used to extract small elements and details from given images. These two types of transforms in which, the top-hat transform is defined as the difference between the input image and its opening by some structuring element, while the black-hat transform is defined as the difference between the closing and the input image. These transforms are used for various image processing tasks, such as feature extraction, background equalization, image enhancement, and others.

Code:

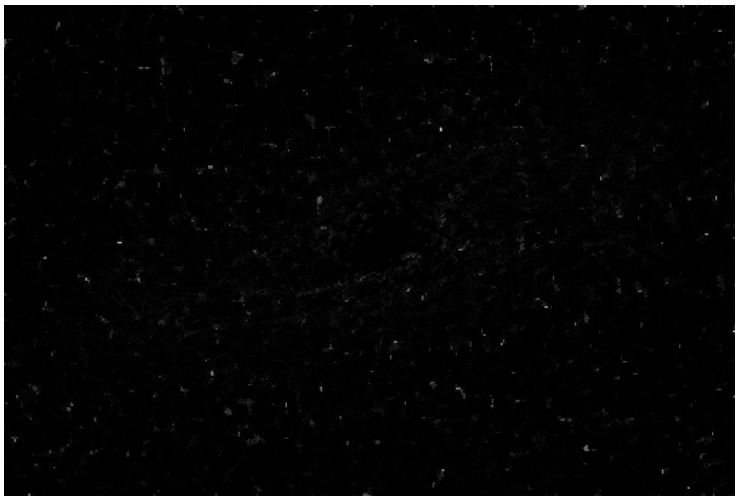
```
import cv2 as cv
import numpy as np
from google.colab.patches import cv2_imshow
img = cv.imread('/content/tophat.png', cv.IMREAD_GRAYSCALE)

kernel = np.ones((5,5),np.uint8)
erosion = cv.erode(img,kernel,iterations = 1)
assert img is not None, "file could not be read, check with
os.path.exists()"
tophat = cv.morphologyEx(img, cv.MORPH_TOPHAT, kernel)
blackhat = cv.morphologyEx(img, cv.MORPH_BLACKHAT, kernel)
cv2_imshow(tophat)
cv2_imshow(blackhat)
```

Input Image:



Output image:

Top Hat



Bottom Hat

Practical No. 9(A)

Aim : Program for Edge detection using Sobel, Prewitt, Marr-Hildreth and Canny

1) Sobel Edge Detection:
   The Sobel operator is a widely used edge detection technique that calculates the gradient magnitude of an image. It is based on convolving the image with two 3x3 kernels, one for detecting edges in the horizontal direction (Sobel_x) and the other for detecting edges in the vertical direction (Sobel_y). The resulting gradient magnitude is computed as the square root of the sum of squared gradients in both directions.

2) Prewitt Edge Detection:
   Similar to the Sobel operator, the Prewitt operator is another gradient-based edge detection technique. It also uses two 3x3 kernels, one for detecting horizontal edges and the other for detecting vertical edges. By convolving the image with these kernels, the gradient magnitude is computed.

3) Marr-Hildreth Edge Detection:
   The Marr-Hildreth operator combines edge detection with the concept of image smoothing using the Laplacian of Gaussian (LoG) filter. It involves convolving the image with a Gaussian filter to smooth it and then computing the Laplacian to detect edges. The resulting edges are often more precise and well-defined compared to other techniques.

4) Canny Edge Detection:
   The Canny edge detection algorithm is a multi-stage process that involves several steps:

   - Noise Reduction :- The image is first smoothed using a Gaussian filter to reduce noise.
   - Gradient Calculation :- The gradient magnitude and direction are computed using the Sobel operator.
   - Non-maximum Suppression :- Only the local maxima in the gradient magnitude are considered as potential edge pixels, suppressing all other non-maximum values.
   - Double Thresholding :- Edges are classified as strong, weak, or non-edges based on two user-defined thresholds.
   - Edge Tracking by Hysteresis :- Weak edges are connected to strong edges to form continuous edges, while non-edges are removed.

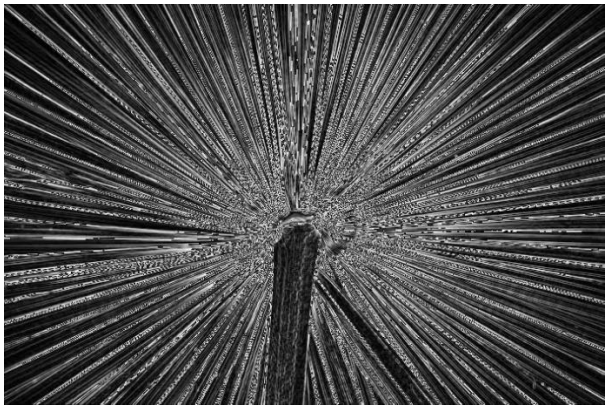
Code:

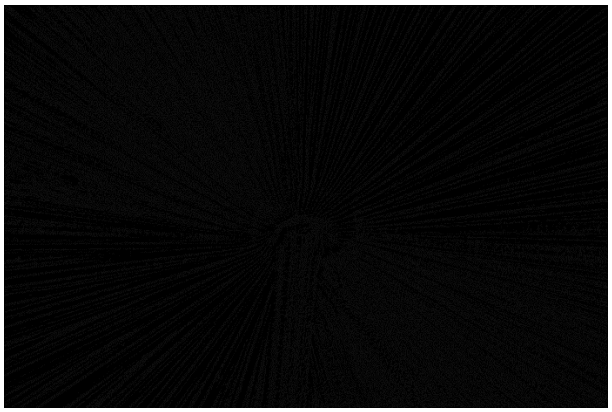
```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Load the image
image = cv2.imread('/content/leaf.jpg', cv2.IMREAD_GRAYSCALE)

# Apply Sobel operator
```

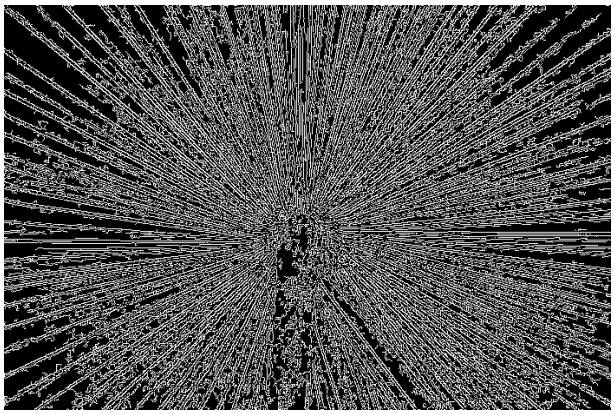
```
sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
sobel = np.sqrt(sobel_x**2 + sobel_y**2)

# Apply Prewitt operator
prewitt_x = cv2.filter2D(image, -1, np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]]))
prewitt_y = cv2.filter2D(image, -1, np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]]))
prewitt = np.sqrt(prewitt_x**2 + prewitt_y**2)

# Apply Marr-Hildreth operator
marr_hildreth = cv2.Laplacian(image, cv2.CV_64F)
marr_hildreth = cv2.Canny(np.uint8(np.abs(marr_hildreth)), 30, 100)

# Apply Canny edge detection
canny = cv2.Canny(image, 30, 100)

# Display the results
cv2_imshow(sobel.astype(np.uint8))
cv2_imshow(prewitt.astype(np.uint8))
cv2_imshow(marr_hildreth)
cv2_imshow(canny)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

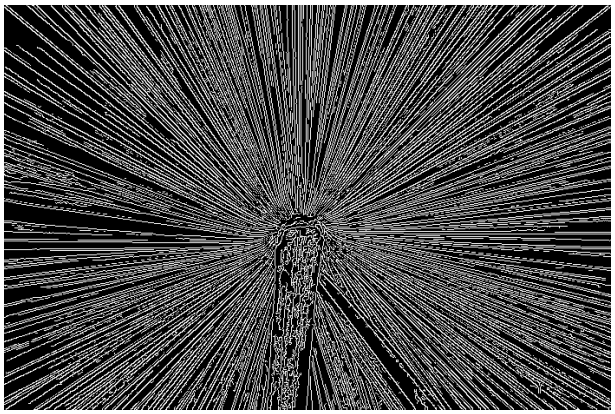
Input:

Output:









Practical No. 9(B)

Aim : Program to illustrate Watershed segmentation algorithm

The Watershed transformation is applied to the gradient image using the generated markers. It treats the intensity values as a topographic surface, where high values represent peaks and low values represent valleys. The markers act as dams or barriers that control the flooding of water in the valleys. The flooding process assigns a unique label to each pixel based on the marker it reaches first.

Code:

```
import cv2
import numpy as np

# Load the image
image = cv2.imread('image.jpg')

# Convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Apply thresholding to create a binary image
_, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV +
cv2.THRESH_OTSU)

# Perform morphological operations to clean up the binary image
kernel = np.ones((3, 3), dtype=np.uint8)
opening = cv2.morphologyEx(binary, cv2.MORPH_OPEN, kernel, iterations=2)
sure_bg = cv2.dilate(opening, kernel, iterations=3)
dist_transform = cv2.distanceTransform(opening, cv2.DIST_L2, 5)
_, sure_fg = cv2.threshold(dist_transform, 0.7 * dist_transform.max(), 255, 0)
sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg, sure_fg)

# Marker labeling
_, markers = cv2.connectedComponents(sure_fg)
markers = markers + 1
markers[unknown == 255] = 0

# Apply Watershed algorithm
markers = cv2.watershed(image, markers)
image[markers == -1] = [0, 0, 255]  # Mark watershed boundaries in red

# Display the result
cv2.imshow('Segmented Image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

Output: