To implement the lab, follow the below steps.

TASK 1

Step 1: Install Docker

Download and install Docker for your operating system. You can find the installation instructions on the Docker website.

To install Docker, you can follow these below steps.

1. Visit the Docker website at https://www.docker.com/ and select the appropriate download for your operating system.
2. Follow the installation instructions for your operating system. The installation process may vary depending on your system, but in general, you will need to run an installer and accept the license agreement.
3. After the installation is complete, you can test that Docker is working by opening a terminal or command prompt and running the command **docker version**. This should display information about the Docker client and server.

Once Docker is installed and working, you can start using it to run containers and manage your applications.

Step 2: Create the Web Application

Create a simple web application using HTML, CSS, JavaScript, and PHP (you can use what you have done for Lab 7). You can use any code editor or IDE of your choice. Save the files in a directory. (Say named lab9)

Note: You can use any two perfectly working web pages from your Lab 7 (Index page and any 2 of the 4 practices)

Step 3: Create a Dockerfile

"A Dockerfile is a text file that contains instructions for building a Docker image."

Create a file named "Dockerfile" in the same "lab9" directory. Copy the following code into the file.

```
FROM php:7.4-apache
COPY . /var/www/html/
EXPOSE 80
```

Below is a brief description of each line in the Dockerfile.

- FROM php:7.4-apache: This line specifies the base image that we will use to build our new image. In this case, we are using the official PHP 7.4 image that comes with the Apache web server.
- COPY . /var/www/html/: This line copies the contents of the current directory (the . symbol) to the /var/www/html/ directory inside the container. This is where Apache looks for files to serve over the web.
- EXPOSE 80: This line tells Docker that our container will be listening on port 80, which is the default port for HTTP traffic (i.e., web traffic).

Step 4: Build the Docker Image

Open a terminal or command prompt and navigate to the "lab9" directory. Run the following command to build the Docker image.

```
docker build -t mywebapp .
```

This command will build a Docker image named "mywebapp" from the Dockerfile in the current directory.

Step 5: Run the Docker Container

Run the following command to start a Docker container from the "mywebapp" image.

```
docker run -d -p 8080:80 --name mywebapp-container mywebapp
```

This command will start a Docker container named "mywebapp-container" from the "mywebapp" image, map port 8080 on the host machine to port 80 in the container and run the container in detached mode.

Step 6: Test the Web Application

Open a web browser and navigate to http://localhost:8080 to test the web application. You should be able to see the web page that you created in step 2.

Step 7: Manage your Docker Image/Container (Optional)

• To stop the container	**docker stop mywebapp-container**
• To start the container	**docker start mywebapp-container**
• To restart the container	**docker restart mywebapp-container**
• To kill the container	**docker kill mywebapp-container**
• To delete the container	**docker rm mywebapp-container**

• To view the docker images	**docker images**
• To view the running containers	**docker ps**
• To view all the containers	**docker ps -a**
• To delete the docker image	**docker rmi mywebapp**

(Note that, here “mywebapp-container” is the name of the container and “mywebapp” is the name of the docker image)

Step 8: Push the Docker Image to a Registry

Sign up for a Docker Hub account if you don't have one already. Run the following commands to push the Docker image to Docker Hub:

```
docker login
docker tag mywebapp username/lab9_docker:tag_name
docker push username/lab9_docker:tag_name
docker logout
```

Make sure to replace “username” with your Docker Hub username and tag_name with a unique tag name for your image.

(Here, “mywebapp” is the name of your docker image, lab9_docker is the name of the repository. You can change it according to yours)

After pushing your Docker image, you will be able to see it on your Docker Hub.

You can also pull the docker images from the Docker Hub registry and run them using the commands below.

```
docker pull <username>/<image-name>:<tag>
docker run -p 8080:80 <username>/<image-name>:<tag>
```

TASK 2

Step 1: Write a C program

For the C program, use any one of the programs you have done for Lab 3. You can use any code editor or IDE of your choice. Save the files in a directory.

Step 2: Create a Dockerfile

Create a file named "Dockerfile" in the same directory. Copy the following code into the file.

```
# Set the base image to use
FROM gcc:latest

# Copy the C program into the container
COPY ScorePossibilities.c .

# Compile the C program
RUN gcc -o ScorePossibilities ScorePossibilities.c

# Set the command to run when the container starts
CMD ["./ScorePossibilities"]
```

Replace ScorePossibilities with your program name.

Step 3: Build the Docker Image

Open a terminal or command prompt and navigate to your directory. Run the following command to build the Docker image.

```
docker build -t lab9_cprogram .
```

This command will build a Docker image named "lab9_cprogram" from the Dockerfile in the current directory.

Step 4: Run the Docker Container

Run the following command to start a Docker container from the "lab9_cprogram" image.

```
docker run -it lab9_cprogram
```

Step 5: Push the Docker Image to a Registry

Follow the steps as given in TASK 1

TASK 3

Submit the Docker hub link of your Docker Images and GitHub repository link (with all the files used for this Lab 9)