Q.1 CODE :

Sobel_Edge_x_Dir.cpp 4

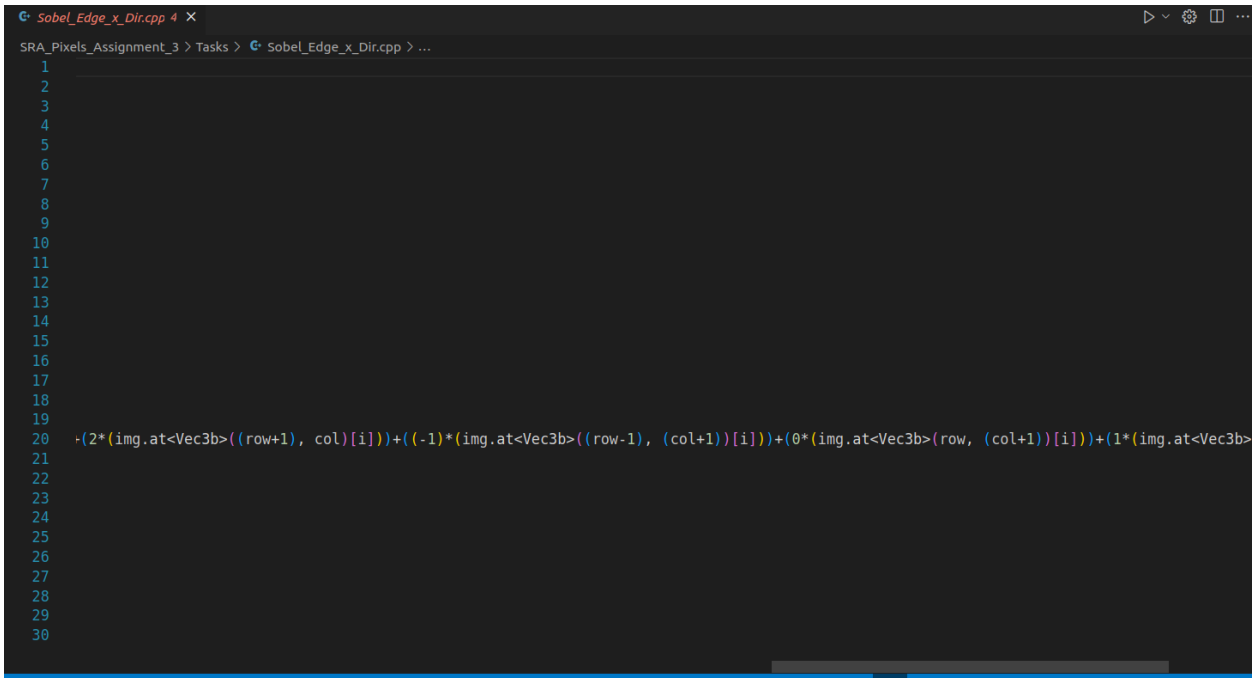
SRA_Pixels_Assignment_3 > Tasks > Sobel_Edge_x_Dir.cpp > ...

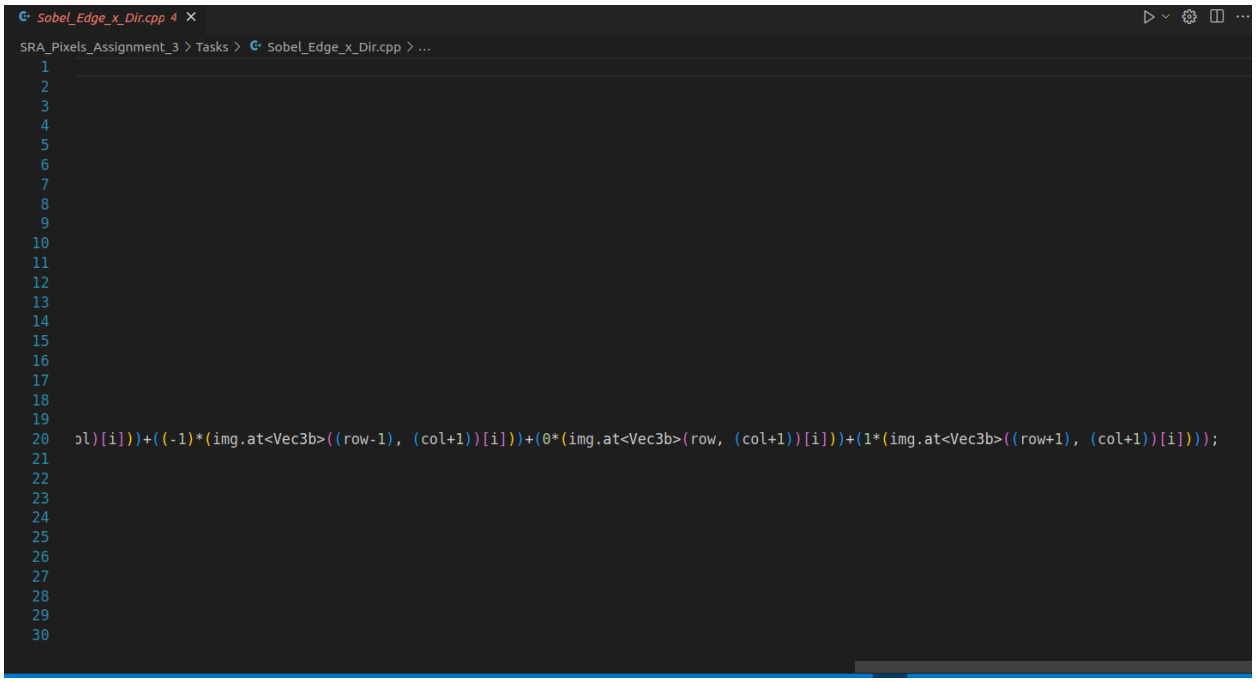
```cpp
#include <iostream>

#include <opencv2/highgui.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/imgproc.hpp>

using namespace std;
using namespace cv;

int main(){
    Mat img;
    img=imread("dog.jpg");
    int img_wid=img.size().width;
    int img_hgt=img.size().height;
    Mat img1((img_hgt-2), (img_wid-2), CV_8UC3, Scalar(0, 0, 0));
    imshow("Original Image", img);
    for(int row=0; row<(img_hgt-2); row++){
        for(int col=0; col<(img_wid-2); col++){
            for(int i=0; i<3; i++){
                img1.at<Vec3b>(row, col)[i]=(((-1)*(img.at<Vec3b>((row-1), (col-1))[i]))+(0*(img.at<Vec3b>(row, (col-1)
                if(img1.at<Vec3b>(row, col)[i]>255){
                    img1.at<Vec3b>(row, col)[i]=255;
                }
            }
        }
    }
    imshow("Processed Image", img1);
    waitKey(0);
    return 0;
}
```

Sobel_Edge_x_Dir.cpp 4

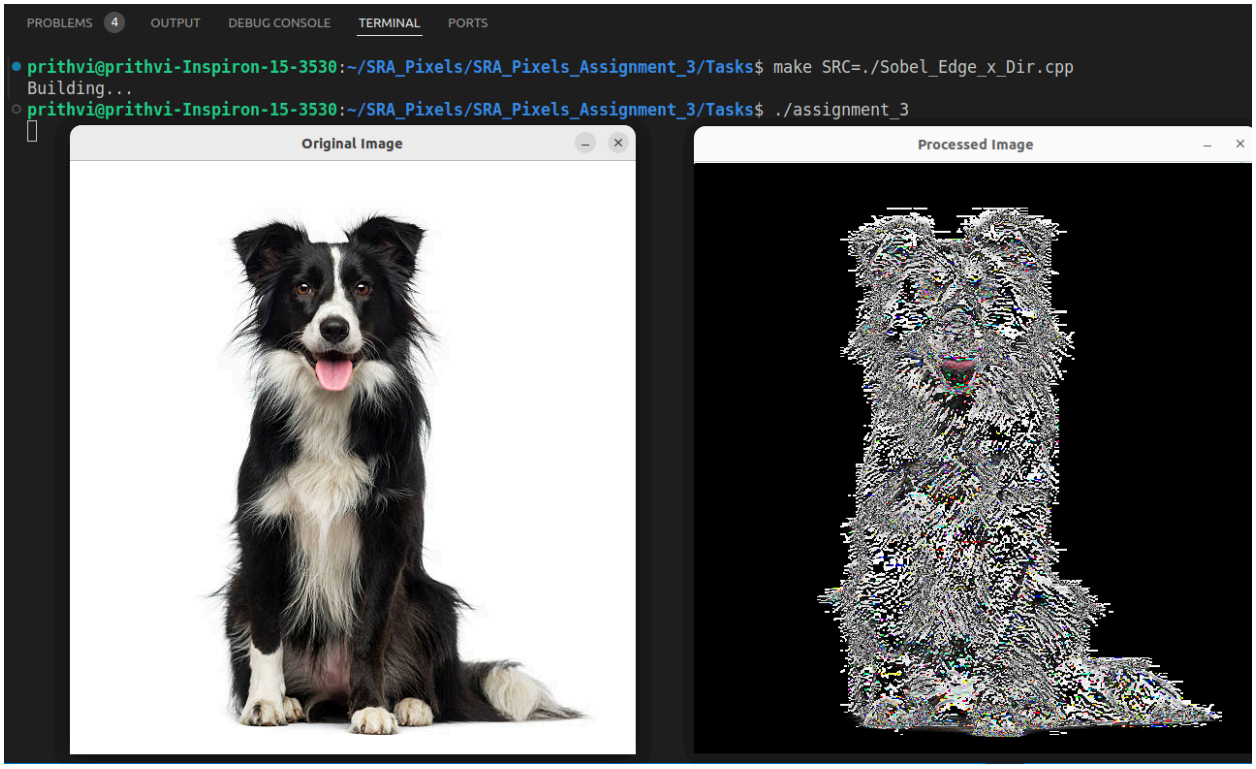
SRA_Pixels_Assignment_3 > Tasks > Sobel_Edge_x_Dir.cpp > ...

```cpp
)+(1*(img.at<Vec3b>((row+1), (col-1))[i]))+((-2)*(img.at<Vec3b>((row-1), col)[i]))+(0*(img.at<Vec3b>(row, col)[i]))+(2*(img.at<Vec3b>((
```


Sobel_Edge_x_Dir.cpp 4
SRA_Pixels_Assignment_3 > Tasks > Sobel_Edge_x_Dir.cpp > ...
(2*(img.at<Vec3b>((row+1), col)[i]))+((-1)*(img.at<Vec3b>((row-1), (col+1))[i]))+(0*(img.at<Vec3b>(row, (col+1))[i]))+(1*(img.at<Vec3b>


Sobel_Edge_x_Dir.cpp 4
SRA_Pixels_Assignment_3 > Tasks > Sobel_Edge_x_Dir.cpp > ...
ɔl)[i]))+((-1)*(img.at<Vec3b>((row-1), (col+1))[i]))+(0*(img.at<Vec3b>(row, (col+1))[i]))+(1*(img.at<Vec3b>((row+1), (col+1))[i])));

Q.1 OUTPUT :

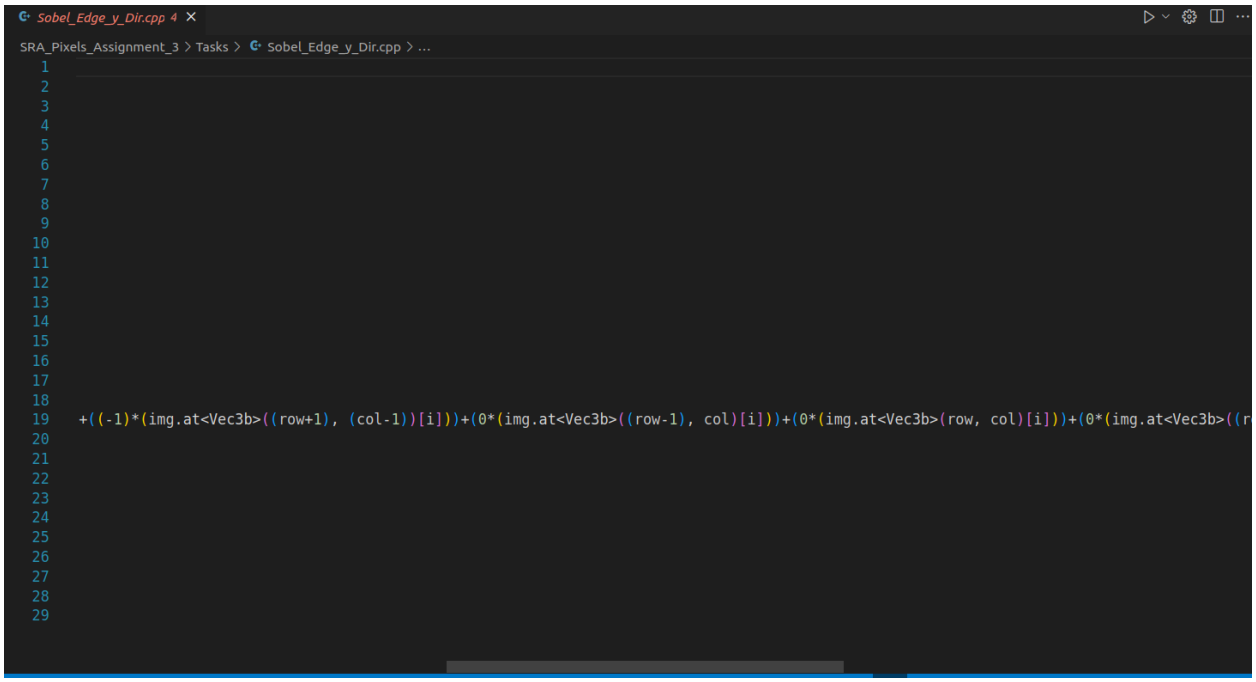


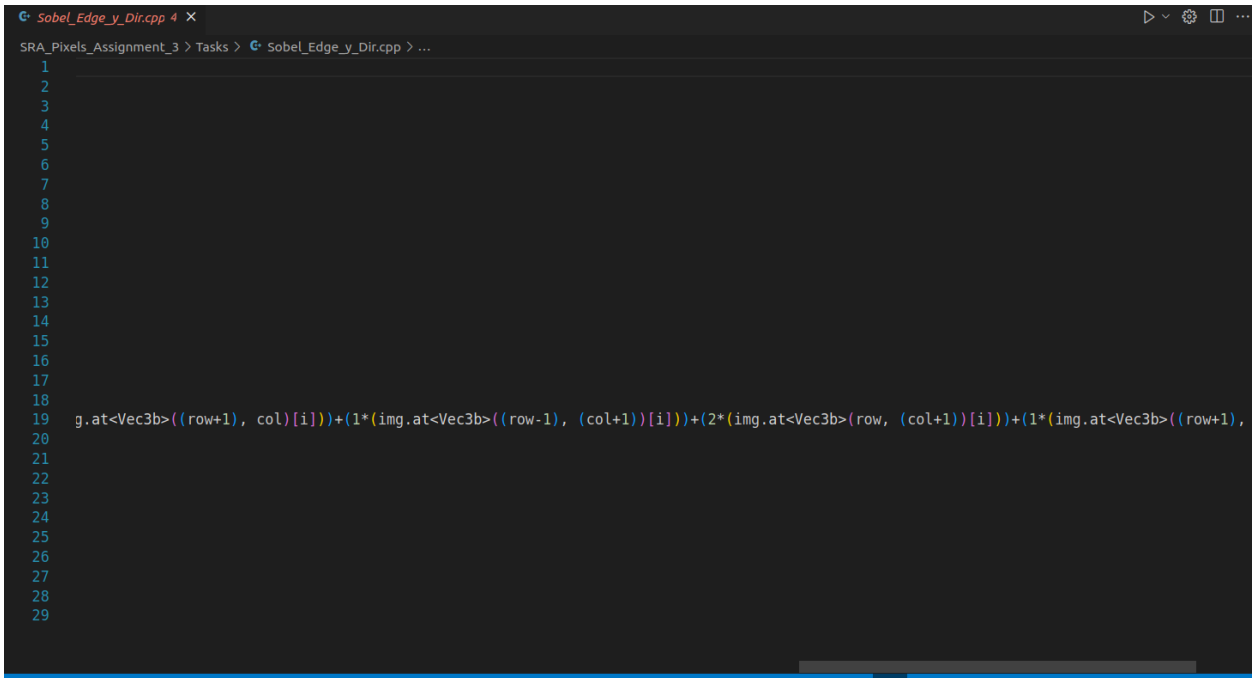
Q.2 CODE :

Sobel_Edge_y_Dir.cpp 4

SRA_Pixels_Assignment_3 > Tasks > Sobel_Edge_y_Dir.cpp > ...

```cpp
#include <iostream>
#include <opencv2/highgui.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/imgproc.hpp>

using namespace std;
using namespace cv;

int main(){
    Mat img;
    img=imread("dog.jpg");
    int img_wid=img.size().width;
    int img_hgt=img.size().height;
    Mat img1((img_hgt-2), (img_wid-2), CV_8UC3, Scalar(0, 0, 0));
    imshow("Original Image", img);
    for(int row=0; row<(img_hgt-2); row++){
        for(int col=0; col<(img_wid-2); col++){
            for(int i=0; i<3; i++){
                img1.at<Vec3b>(row, col)[i]=(((-1)*(img.at<Vec3b>((row-1), (col-1))[i]))+((-2)*(img.at<Vec3b>(row, (col
                if(img1.at<Vec3b>(row, col)[i]>255){
                    img1.at<Vec3b>(row, col)[i]=255;
                }
            }
        }
    }
    imshow("Processed Image", img1);
    waitKey(0);
    return 0;
}
```


Sobel_Edge_y_Dir.cpp 4
SRA_Pixels_Assignment_3 > Tasks > Sobel_Edge_y_Dir.cpp > ...
+((-1)*(img.at<Vec3b>((row+1), (col-1))[i]))+(0*(img.at<Vec3b>((row-1), col)[i]))+(0*(img.at<Vec3b>(row, col)[i]))+(0*(img.at<Vec3b>((r

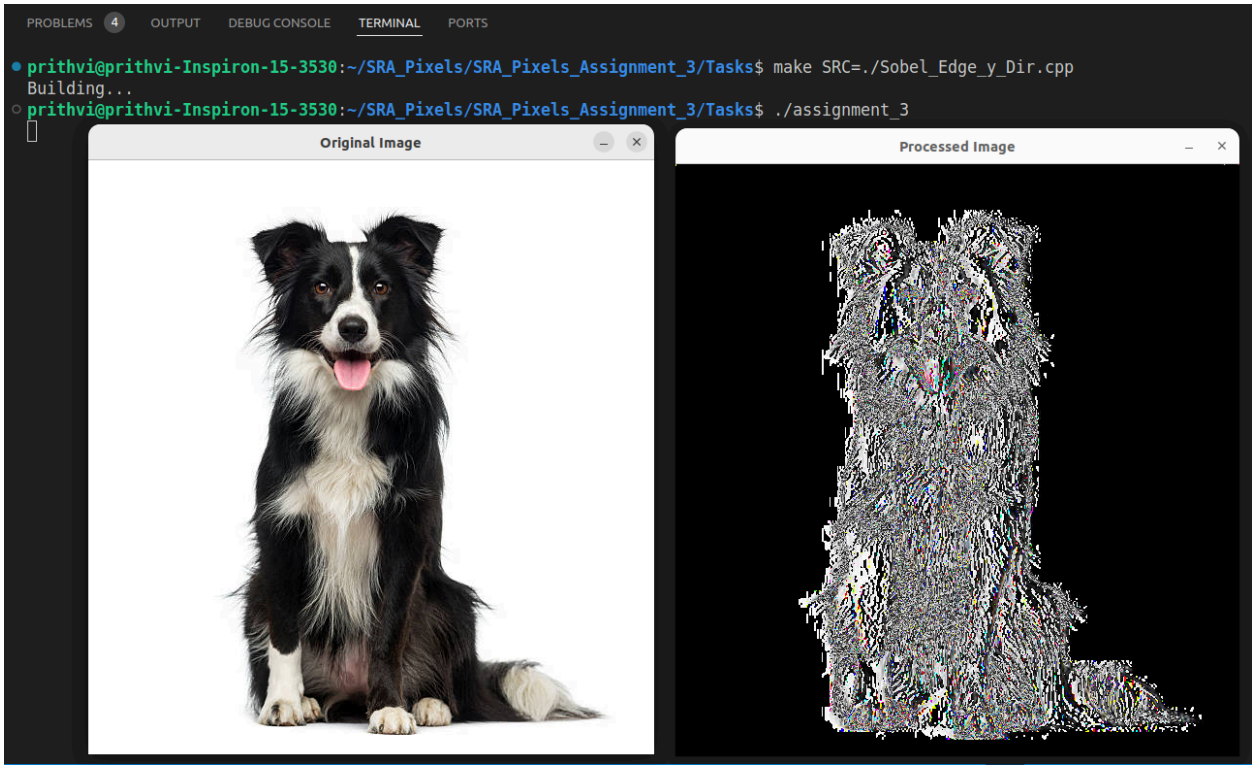

Sobel_Edge_y_Dir.cpp 4
SRA_Pixels_Assignment_3 > Tasks > Sobel_Edge_y_Dir.cpp > ...
g.at<Vec3b>((row+1), col)[i]))+(1*(img.at<Vec3b>((row-1), (col+1))[i]))+(2*(img.at<Vec3b>(row, (col+1))[i]))+(1*(img.at<Vec3b>((row+1),

Sobel_Edge_y_Dir.cpp 4

SRA_Pixels_Assignment_3 > Tasks > Sobel_Edge_y_Dir.cpp > ...

```
, col)[i]))+(1*(img.at<Vec3b>((row-1), (col+1))[i]))+(2*(img.at<Vec3b>(row, (col+1))[i]))+(1*(img.at<Vec3b>((row+1), (col+1))[i])));
```

Q.2 OUTPUT :



Q.3 CODE :

Guassian_Blur.cpp 4

SRA_Pixels_Assignment_3 > Tasks > Guassian_Blur.cpp > ...

```cpp
#include <iostream>
#include <opencv2/highgui.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/imgproc.hpp>

using namespace std;
using namespace cv;

int main(){
    Mat img;
    img=imread("waterfall.jpg");
    int img_wid=img.size().width;
    int img_hgt=img.size().height;
    Mat img_gauss((img_hgt-2), (img_wid-2), CV_8UC3, Scalar(0, 0, 0));
    imshow("Original Image", img);
    int y=0;
    {
        loop1:for(int row=0; row<(img_hgt-2); row++){
            for(int col=0; col<(img_wid-2); col++){
                for(int i=0; i<3; i++){
                    img_gauss.at<Vec3b>(row, col)[i]=((((1*(img.at<Vec3b>((row-1), (col-1))[i]))+(2*(img.at<Vec3b>(row,
                    if(img_gauss.at<Vec3b>(row, col)[i]>255){
                        img_gauss.at<Vec3b>(row, col)[i]=255;
                    }
                }
            }
        }
    }
    imshow("Processed Image", img_gauss);
    waitKey(0);
    return 0;
}
```

Guassian_Blur.cpp 4

SRA_Pixels_Assignment_3 > Tasks > Guassian_Blur.cpp > ...

```cpp
)[i]))+(1*(img.at<Vec3b>((row+1), (col-1))[i]))+(2*(img.at<Vec3b>((row-1), col)[i]))+(4*(img.at<Vec3b>(row, col)[i]))+(
```

Guassian_Blur.cpp 4

SRA_Pixels_Assignment_3 > Tasks > Guassian_Blur.cpp > ...

```
c3b>((row+1), col)[i]))+(1*(img.at<Vec3b>((row-1), (col+1))[i]))+(2*(img.at<Vec3b>(row, (col+1))[i]))+(1*(img.at<Vec3b>
```

Guassian_Blur.cpp 4

SRA_Pixels_Assignment_3 > Tasks > Guassian_Blur.cpp > ...

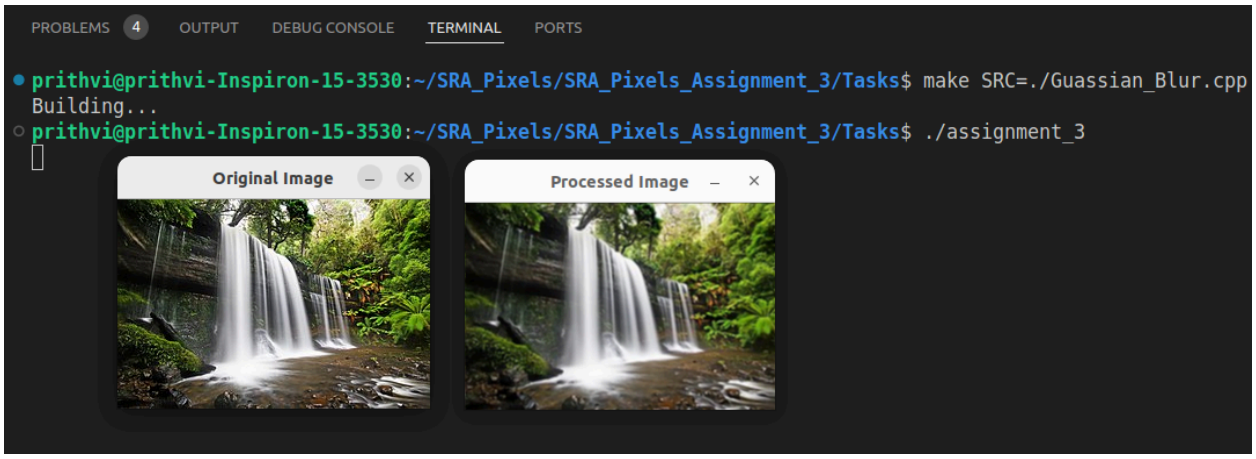
```
)[i]))+(1*(img.at<Vec3b>((row-1), (col+1))[i]))+(2*(img.at<Vec3b>(row, (col+1))[i]))+(1*(img.at<Vec3b>((row+1), (col+1)
```

Guassian_Blur.cpp 4

SRA_Pixels_Assignment_3 > Tasks > Guassian_Blur.cpp > ...

```cpp
int main(){
    Mat img;
    img=imread("waterfall.jpg");
    int img_wid=img.size().width;
    int img_hgt=img.size().height;
    Mat img_gauss((img_hgt-2), (img_wid-2), CV_8UC3, Scalar(0, 0, 0));
    imshow("Original Image", img);
    int y=0;
    {
        loop1:for(int row=0; row<(img_hgt-2); row++){
            for(int col=0; col<(img_wid-2); col++){
                for(int i=0; i<3; i++){
                    img_gauss.at<Vec3b>(row, col)[i]=((((1*(img.at<Vec3b>((row-1), (col-1))[i]))+(2*(img.at<Vec3b>(row,
                    if(img_gauss.at<Vec3b>(row, col)[i]>255){
                        img_gauss.at<Vec3b>(row, col)[i]=255;
                    }
                }
            }
        }
    }
    imshow("Processed Image", img_gauss);
    waitKey(0);
    return 0;
}
```

Q.3 OUTPUT :



Q.4 CODE :

Sharpen.cpp 4

SRA_Pixels_Assignment_3 > Tasks > Sharpen.cpp > ...

```cpp
#include <iostream>
#include <opencv2/highgui.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/imgproc.hpp>

using namespace std;
using namespace cv;

int main(){
    Mat img;
    img=imread("house.jpg");
    int img_wid=img.size().width;
    int img_hgt=img.size().height;
    Mat img_sharp((img_hgt-2), (img_wid-2), CV_8UC3, Scalar(0, 0, 0));
    imshow("Original Image", img);
    int y=0;
    {
        loop1:for(int row=0; row<(img_hgt-2); row++){
            for(int col=0; col<(img_wid-2); col++){
                for(int i=0; i<3; i++){
                    img_sharp.at<Vec3b>(row, col)[i]=((0*(img.at<Vec3b>((row-1), (col-1))[i]))+((-0.5)*(img.at<Vec3b>(row, (col-1))[i])
                    if(img_sharp.at<Vec3b>(row, col)[i]>255){
                        img_sharp.at<Vec3b>(row, col)[i]=255;
                    }
                }
            }
        }
    }
    imshow("Processed Image", img_sharp);
    waitKey(0);
    return 0;
}
```

Sharpen.cpp 4

SRA_Pixels_Assignment_3 > Tasks > Sharpen.cpp > ...

```cpp
1))[i]))+(0*(img.at<Vec3b>((row+1), (col-1))[i]))+((-0.5)*(img.at<Vec3b>((row-1), col)[i]))+(3*(img.at<Vec3b>(row, col)[i]))+((-0.5)*(i
```

Sharpen.cpp 4

SRA_Pixels_Assignment_3 > Tasks > Sharpen.cpp > ...

```
))+((-0.5)*(img.at<Vec3b>((row+1), col)[i]))+(0*(img.at<Vec3b>((row-1), (col+1))[i]))+((-0.5)*(img.at<Vec3b>(row, (col+1))[i]))+(0*(img
```

Sharpen.cpp 4

SRA_Pixels_Assignment_3 > Tasks > Sharpen.cpp > ...

```
)[i]))+(0*(img.at<Vec3b>((row-1), (col+1))[i]))+((-0.5)*(img.at<Vec3b>(row, (col+1))[i]))+(0*(img.at<Vec3b>((row+1), (col+1))[i])));
```

Sharpen.cpp 4

SRA_Pixels_Assignment_3 > Tasks > Sharpen.cpp > ...

```
 9  int main(){
17      {
18          loop1:for(int row=0; row<(img_hgt-2); row++){
19              for(int col=0; col<(img_wid-2); col++){
20                  for(int i=0; i<3; i++){
21                      img_sharp.at<Vec3b>(row, col)[i]=((0*(img.at<Vec3b>((row-1), (col-1))[i]))+((-0.5)*(img.at<Vec3b>(row, (col-1))[i])
22                      if(img_sharp.at<Vec3b>(row, col)[i]>255){
23                          img_sharp.at<Vec3b>(row, col)[i]=255;
24                      }
25                  }
26              }
27          }
28      }
29      imshow("Processed Image", img_sharp);
30      waitKey(0);
31      return 0;
32  }
```

Q.4 OUTPUT :