Université Euromed De Fès

Ecole D'ingénierie Digitale Et

D’Intelligence Artificielle

<u>Report:</u>

Hsla Image

Done by :

-souha hmamed

-Fatima-Ezzahra Kharbouch

Table of contents

INTRODUCTION:

Images are generally known as a group of pixels represented by the RGB color system, or we have another representation that is a human perception of colors.

In this lab we are using the human perception that is consisting of the HSL color system (Hue, Saturation, Luminance), where the Hue indicates the color, Saturation the degree that differentiate the hue from a neutral gray and lastly Luminance that stands for the level of illumination. The purpose of our report is to see the impact of changing each of these parameters on the image

IMAGE CLASS

We created the image class that inherits from the PNG class, and that contain the following methods:

1- Image (string filename);

2-void lighten(double amount=0.1);

3-void saturate (double amount=0.1);

4-void rotateColor(double angle);

We implemented the methodes :

First of all we need to go through each pixel of the image:

```
for(unsigned i=0; i<width();i++)
        for(unsigned j=0;j < height(); j++)
        {
            //obtenir le pixel  dans la position(i,j)
            HSLAPixel &P = getPixel(i,j);
```

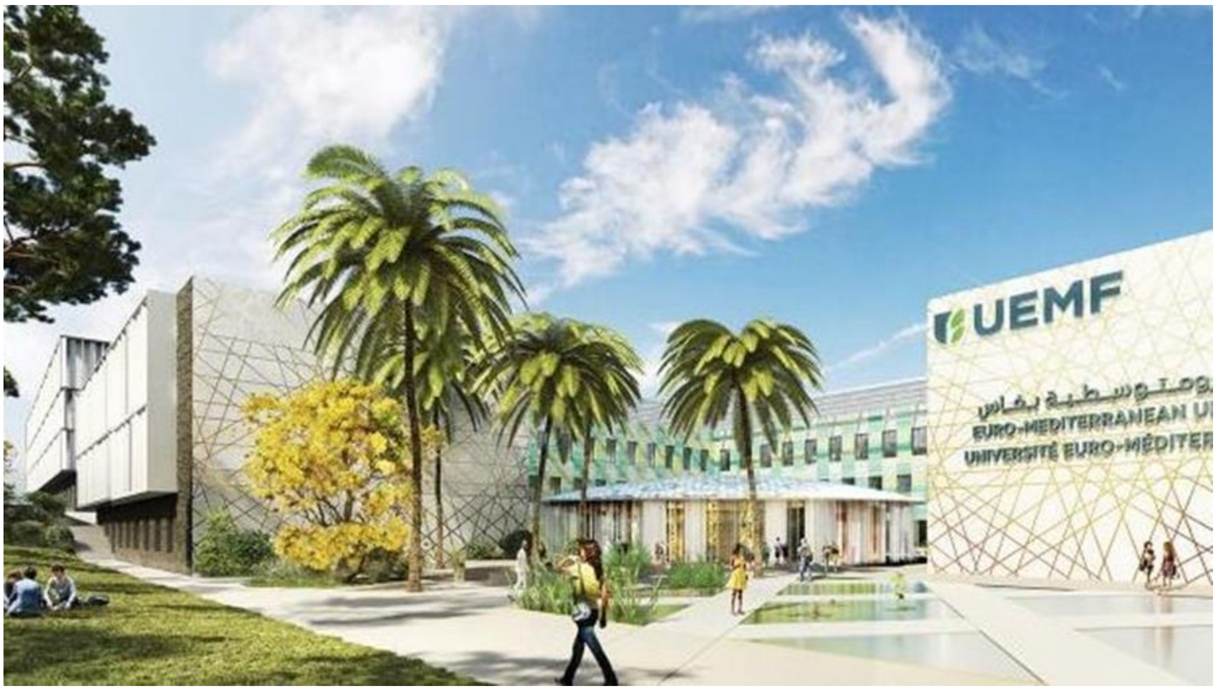
1- Image (string filename): constructor that loads the image from the given filename

2-void lighten(double amount=0.1): change the luminance of each pixel by amount

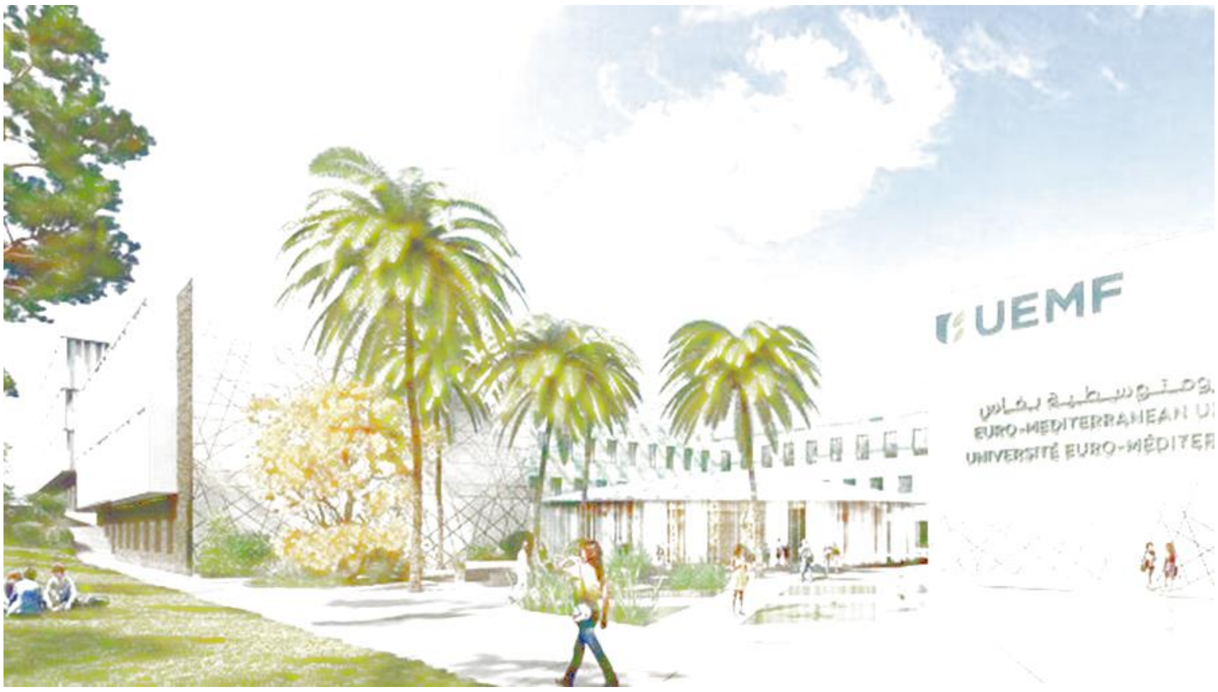
While running through the pixels of the image, we will add the demanded amount of luminance that should be between [0,1]to each pixel

```
P.l +=amount;
P.l= (P.l>0) ? P.l :0;
P.l= (P.l<=1) ? P.l :1;
```

Input:



Output:

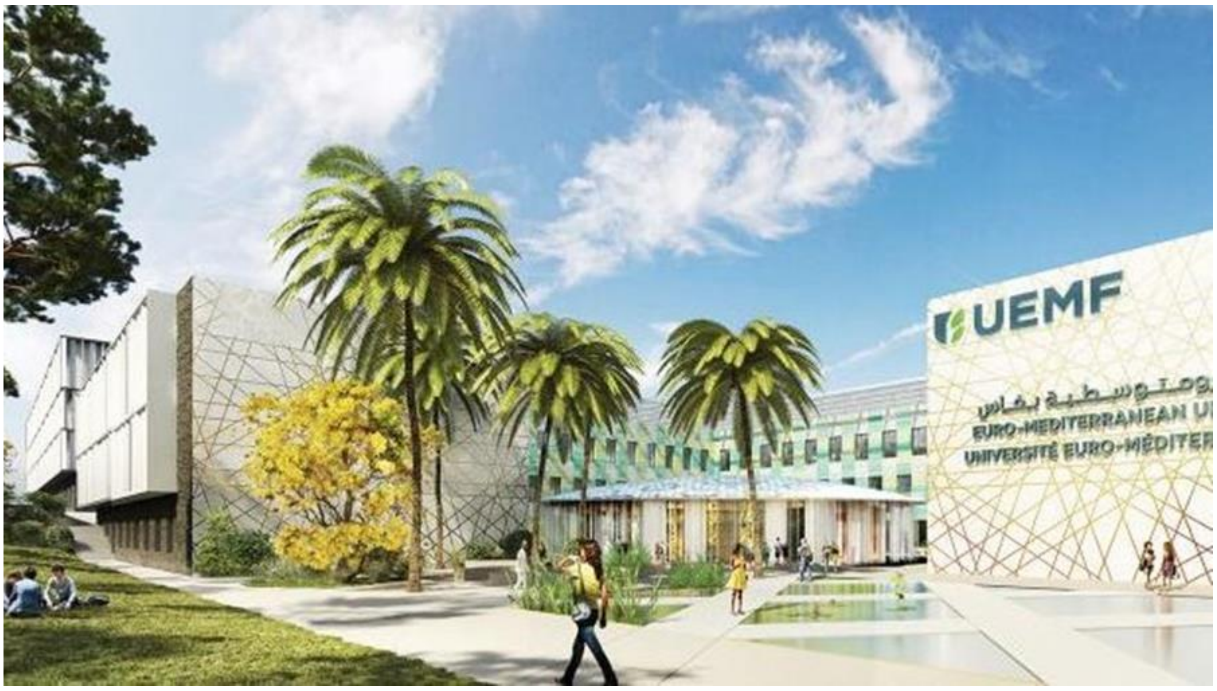


3-void saturate (double amount=0.1) : change the luminance by amount

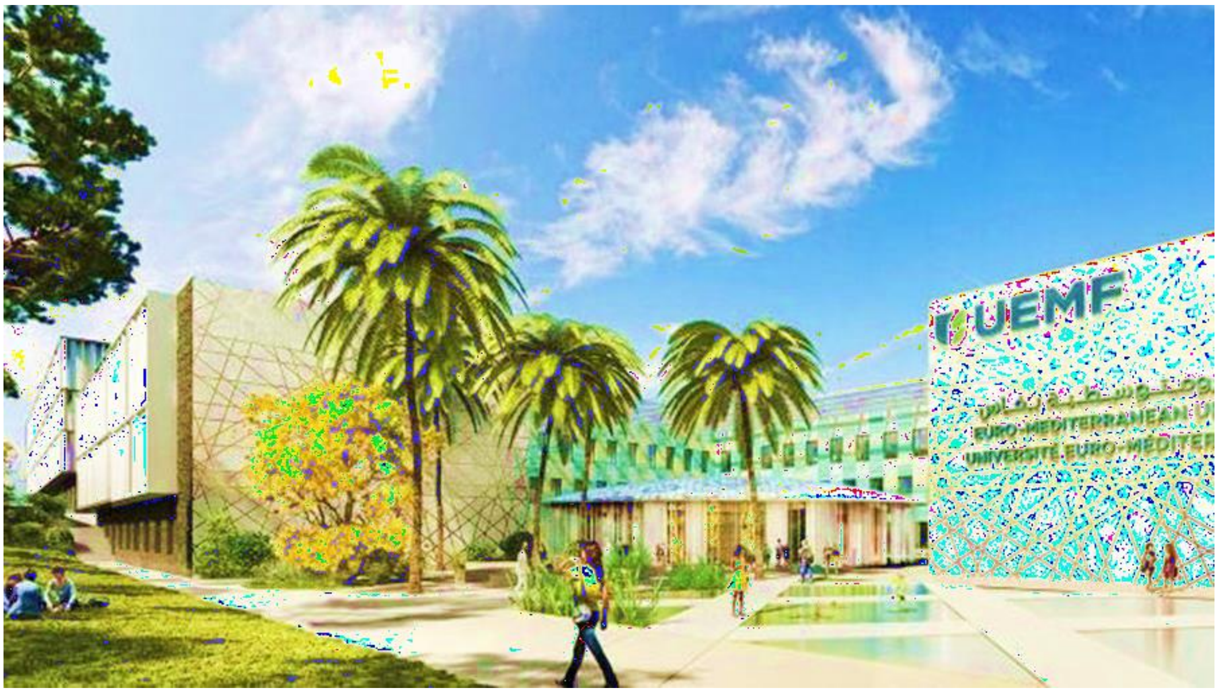
We need to make sure that the luminance will not go beyond [0,1] in every pixel

```
P.s +=amount;
P.s= (P.l>0) ? P.s :0;
P.s= (P.l<=1) ? P.s :1;
```

Input:



Output:



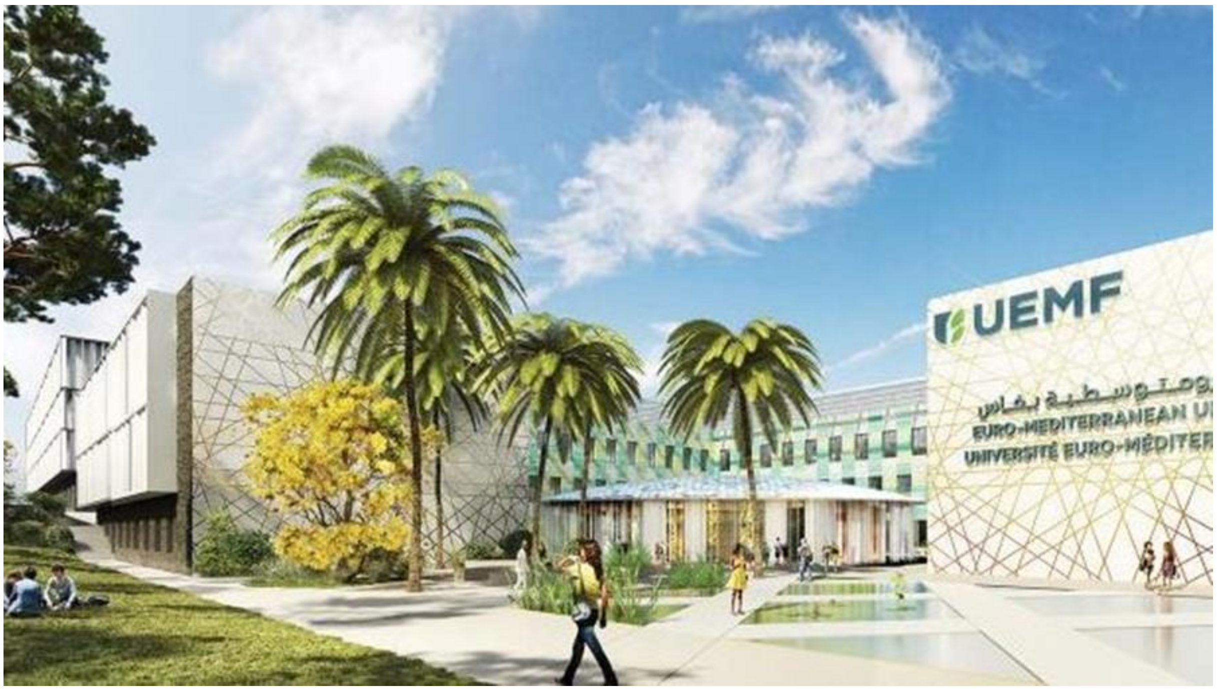
4-void rotateColor(double angle): a method that will allow us to add the value of the angle to every single pixel

```
  P.h+=angle;
```

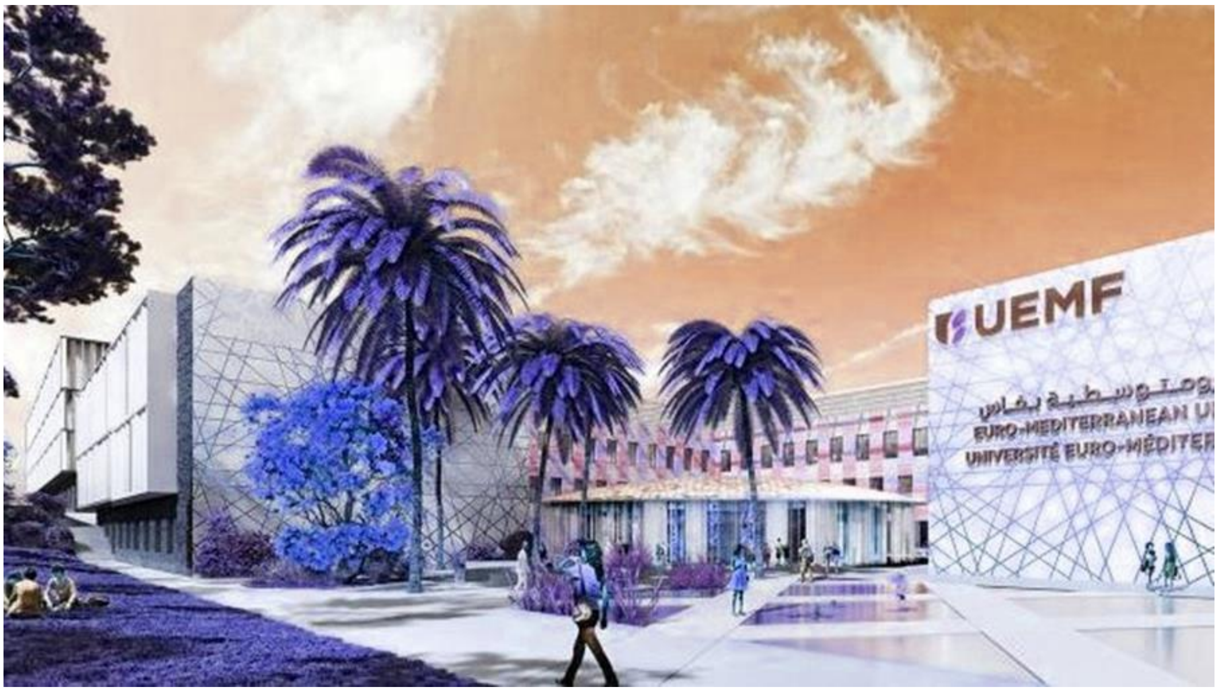
We know that the value of an color represent a cyclical value [0,360], so if this value in greater than 360 or less than 0 we need to respectively subtract or add 360

```
while(P.h>360){
                P.h-=360;}
            while(P.h<0){
                P.h+=360;
    }
```

Input:



Output:



GRAYSCALE CLASS:

We created a class named grayscale whose purpose is to represent the image using only grayscale level

To use this class, we first inherited from the image class

The method used:

1-grayscale(stringfilename): constructor that loads the image from the given filename

First of all, we need to go through each pixel of the image that is why we used the following code:

```
for(unsigned i=0; i<width();i++)
        for(unsigned j=0;j < height(); j++)
```

And to be able to change the saturation of each pixel we need to have the exact position of each one:

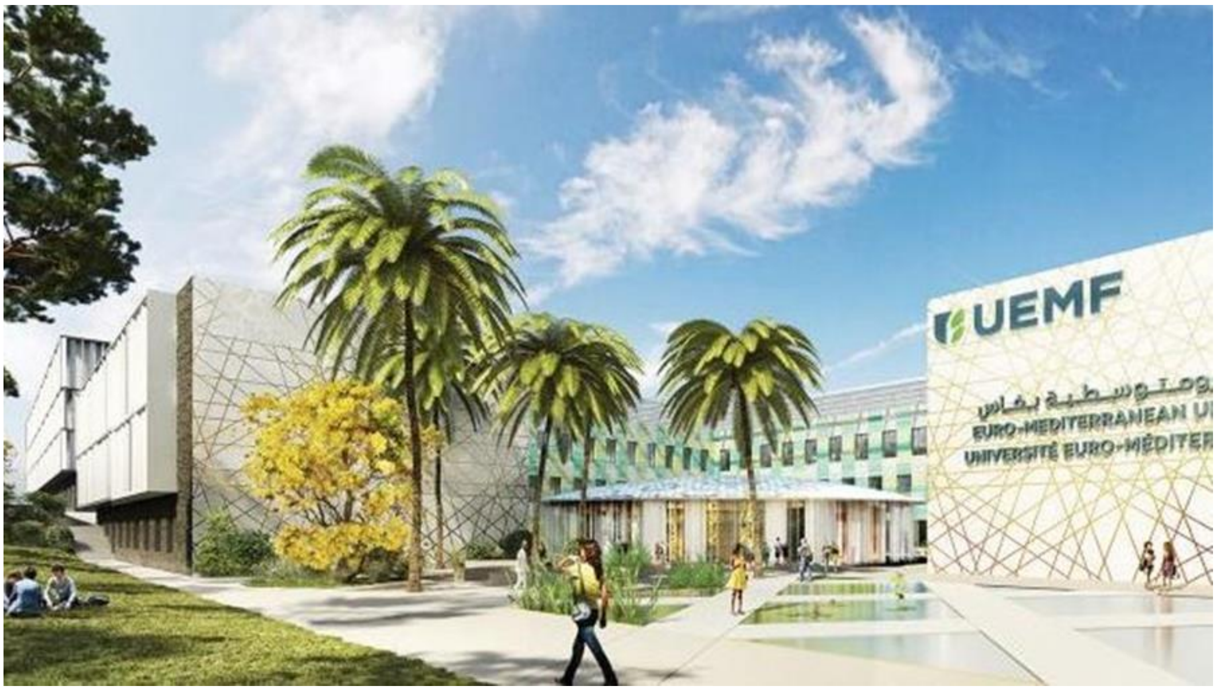
//get the pixel in the position(i,j)

```
HSLAPixel &P = getPixel(i,j);
```

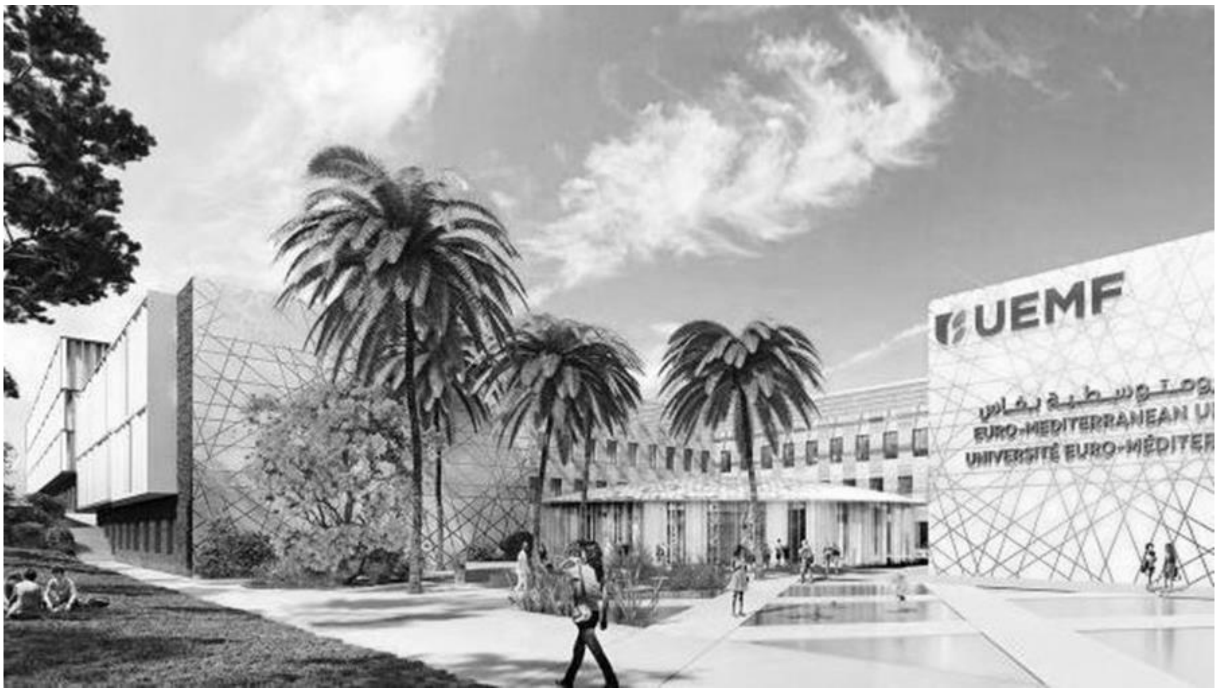
Now to transform the image into a grayscale we need to reduce the saturation to its minimum, knowing that the saturation value run from 0% to 100% the code will be:

```
P.s=0;
```

Input:



output:



ILLINI CLASS

The objective of this class is to change the color of a given image to only two colors.

We created a class named illini that inherits from the class image and contain two public attributes of type int:

```
 int color1=11;
 int color2=216;
```

 and a constructor:

```
illini(string filename, int color1=11, int color2=216);
```

knowing that we have two main colors that the image should change to so we first need to compare the color of the pixel with the given colors and choose the one that is close to the color1 or color2 to know which one to change it with.

To compare the color of the pixel with color 1:

```
if(P.h>color1){
if(P.h-color1<360+color1-P.h)
    a=P.h-color1;
else
    a=360+color1-P.h;

            }
            else{
                if(-P.h+color1<360+color1+P.h)
                    a=-P.h+color1;
                else
                    a=360-color1+P.h;

      }
```

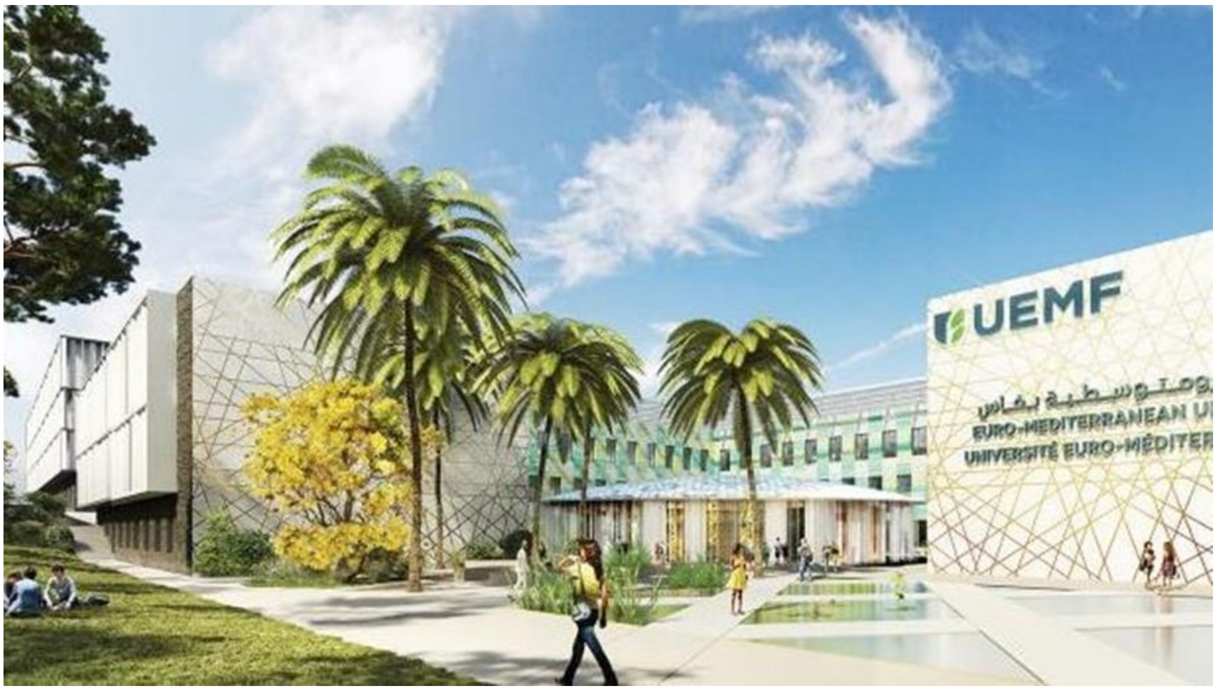
To compare the color of the pixel to color2:

```
if(P.h>color2){
if(P.h-color2<360+color2-P.h)
    b=P.h-color2;
else
    b=360+color1-P.h;

            }
            else{
                if(-P.h+color2<360+color2+P.h)
                    b=-P.h+color2;
                else
                    b=360-color2+P.h;

      }
```

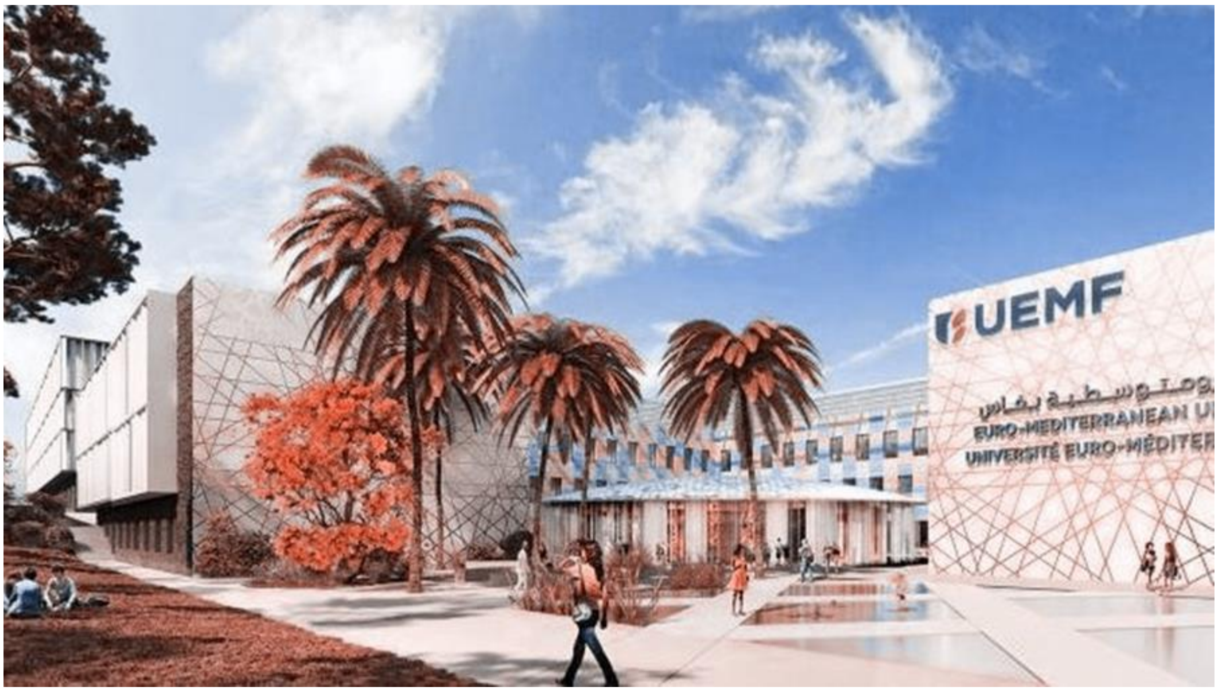
To see if the color of the pixel should be changed to color1 or color2:

```
       if(a<b)
                P.h=color1;
            else
                P.h=color2;
      }
```

Input:



Output:



SPOTLIGHT CLASS

in this part of the project, we will create a class named spotlight that will inherit from image

this class will reduce the luminance of the pixels expect the spotlight that is centered at a given point

the spotlight class contains:

-two private attributes

- **spotlight**(string file, int xCenter,int yCenter): **a constructor that accepts two values**

- void **changeSpotPoint**(int xCenter,int yCenter): **method to change the position of the spotlight.**

To know the number of pixels that are away from the center we use the following code

```
double dis= sqrt((i-xCenter)*(i-xCenter)+(j-yCenter)*(j-yCenter)) ;
```

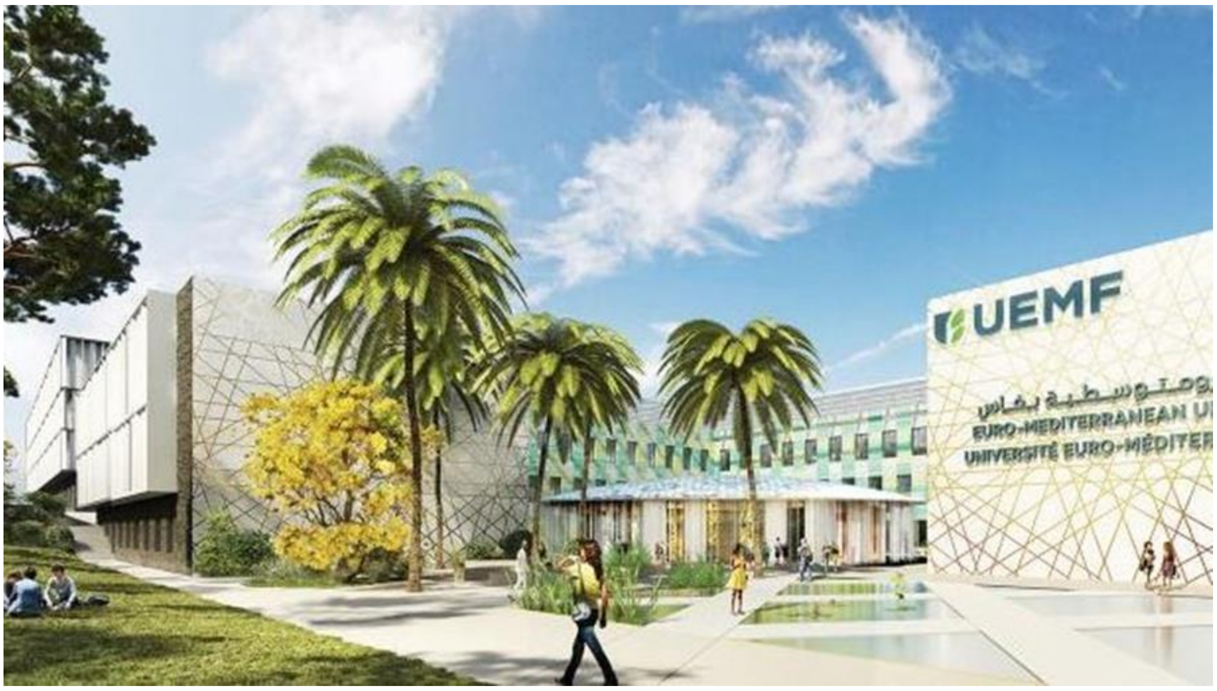
when the distance between the pixel and the center become grater than 160 the luminance will decrease by 0.2 from the original value

```
if(dis>=160){
                    P.l=0.2*P.l;
            }
```

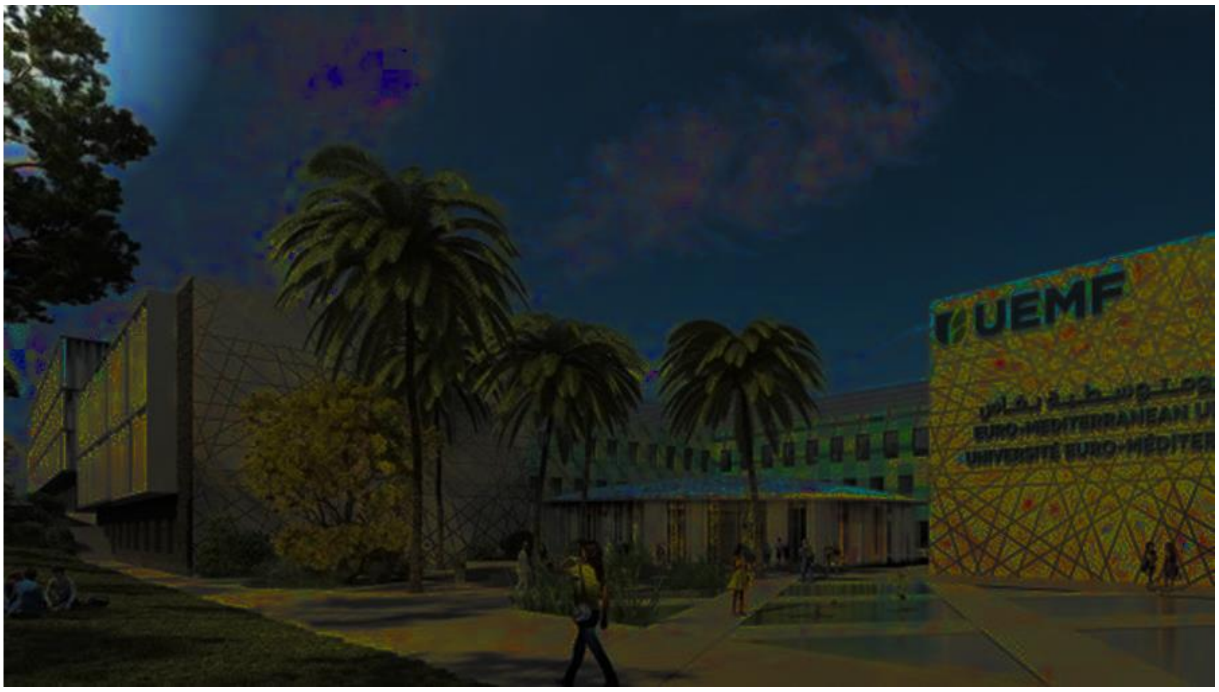
Or in the opposite case the luminance will be

```
else{
              P.l=abs(P.l-dis*0.005*P.l);
            }
```

Input:



Output:


UEMF
EURO-MEDITERRANEAN U
UNIVERSITÉ EURO-MÉDITER