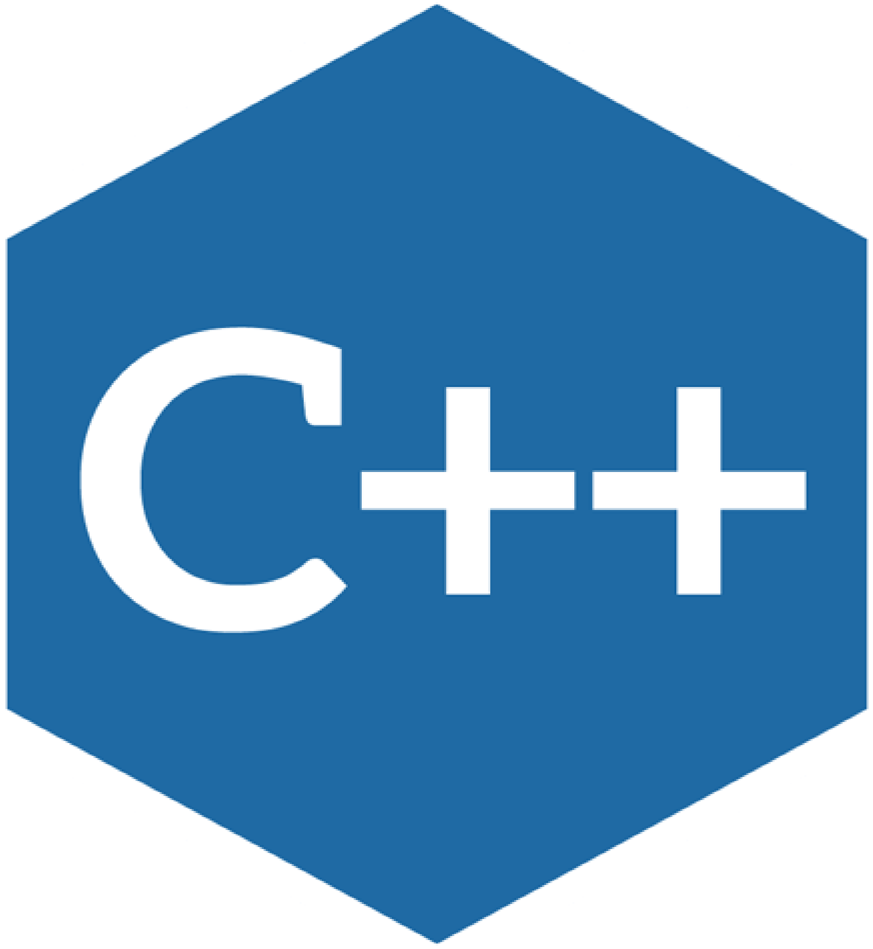
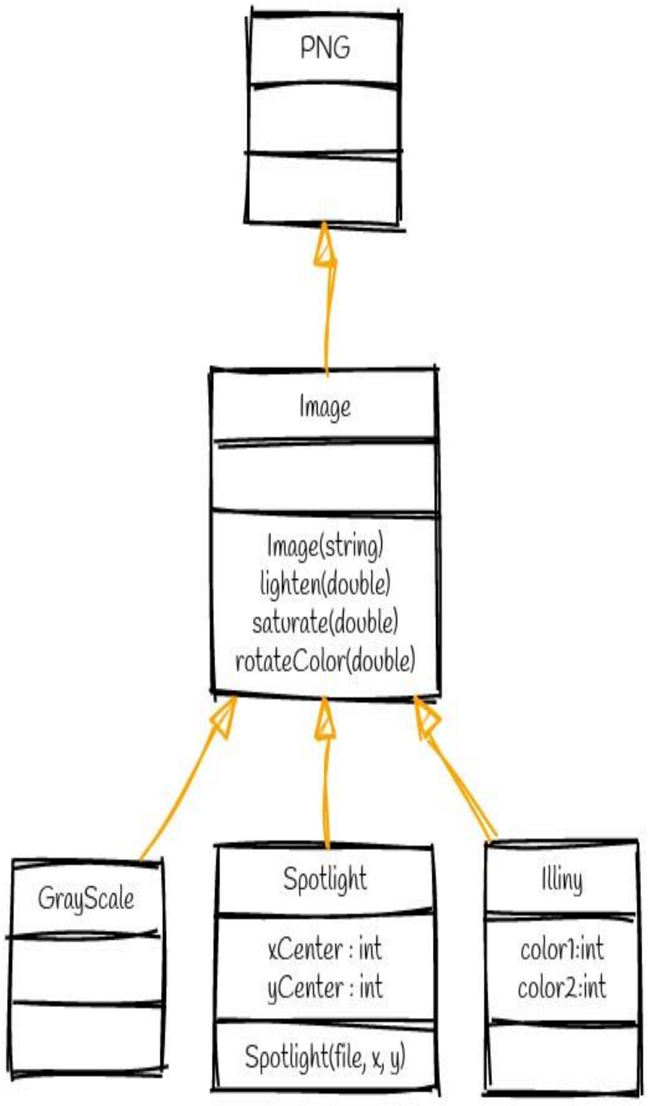
HSLA Images

APPLYING INHERITANCE KNOWLEDGE TO MANIPULATE HSLA IMAGES



Rachid sakassa/badr-eddine benzemroun | project report | 24/10

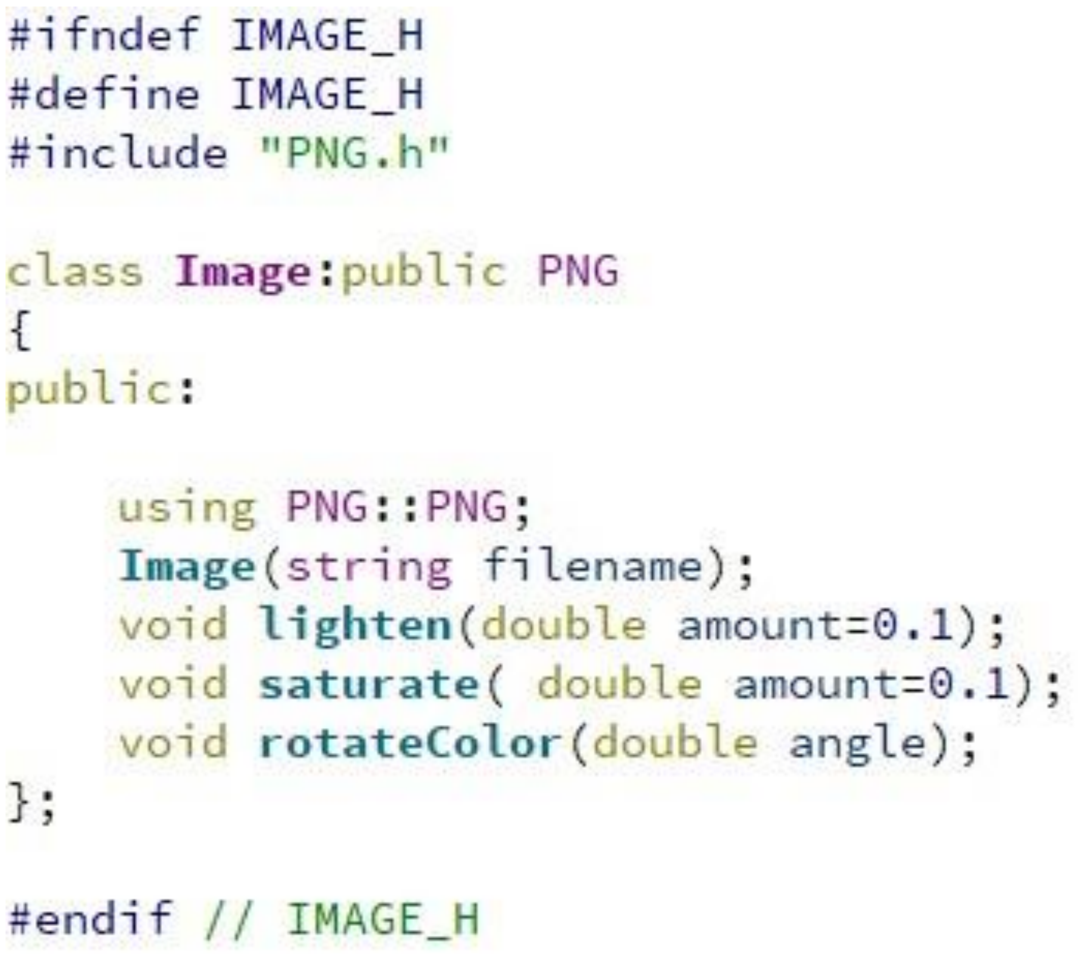
Inhertance diagram



UML class diagram for the additional Images classes.

Image class :

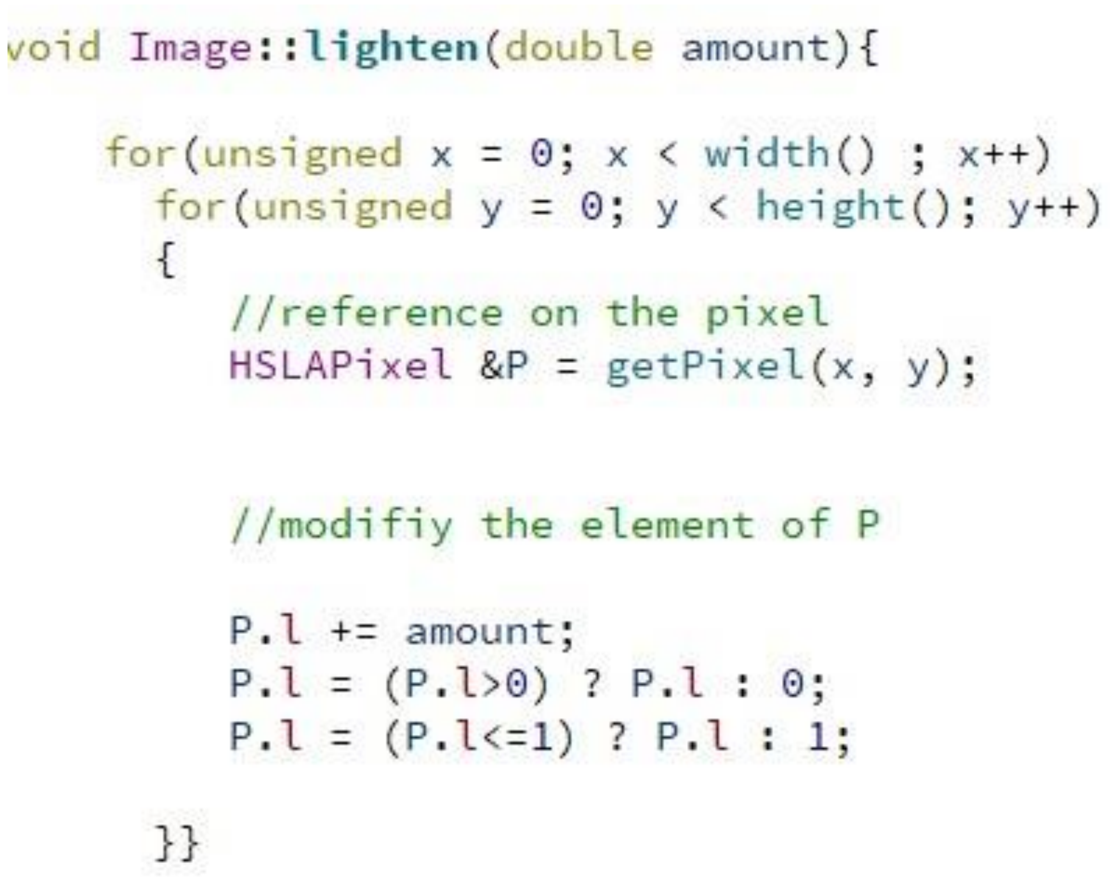
In the header file we declared the constructor and the methods that will be implemented later :



```cpp
#ifndef IMAGE_H
#define IMAGE_H
#include "PNG.h"

class Image:public PNG
{
public:

    using PNG::PNG;
    Image(string filename);
    void lighten(double amount=0.1);
    void saturate( double amount=0.1);
    void rotateColor(double angle);
};

#endif // IMAGE_H
```

Lighten(double amount) changes the luminance of each pixel by amount.

- The function must ensures that luminance remains in the range [0,1]

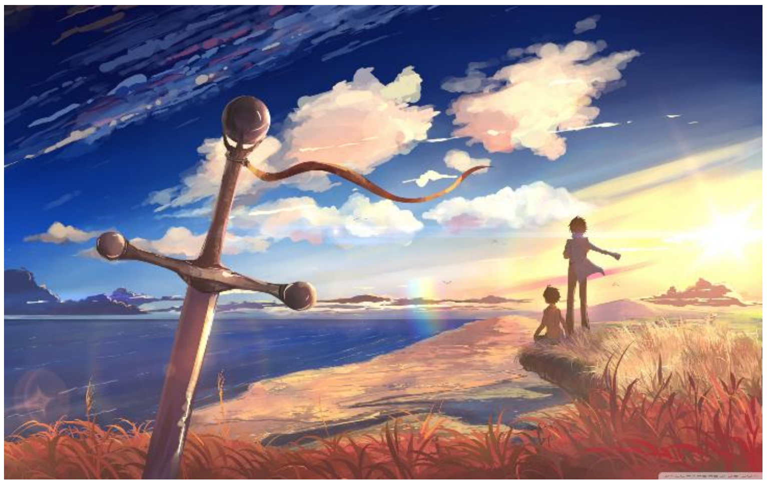


```
void Image::lighten(double amount){

  for(unsigned x = 0; x < width() ; x++)
    for(unsigned y = 0; y < height(); y++)
    {
      //reference on the pixel
      HSLAPixel &P = getPixel(x, y);


      //modifiy the element of P

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

    }}
```

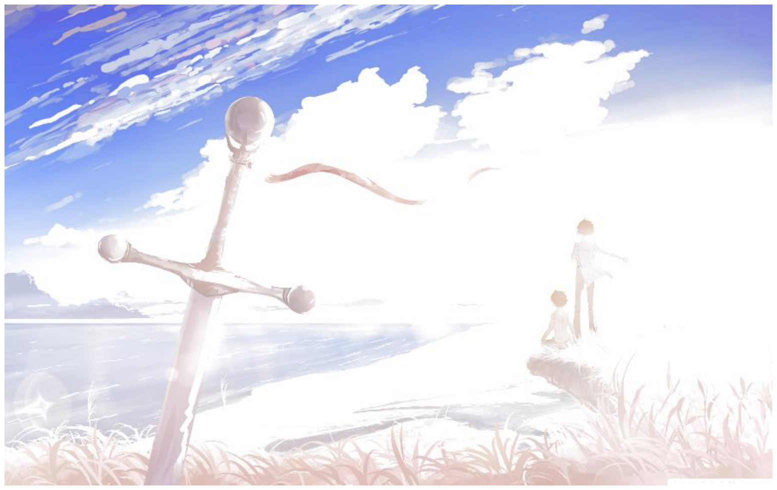
Results

Changing the luminance of the image using lighten(0.5) :

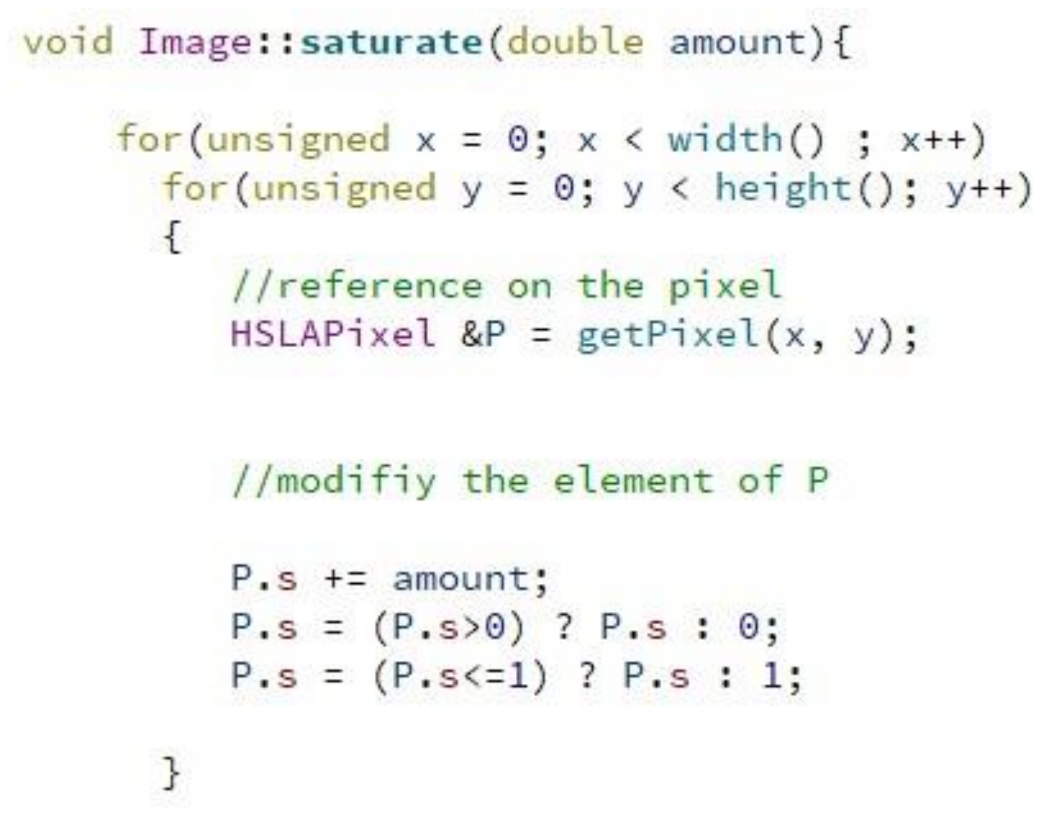
Input:



Output:



`-saturate` changes the intensity of the color of each pixel by an amount.

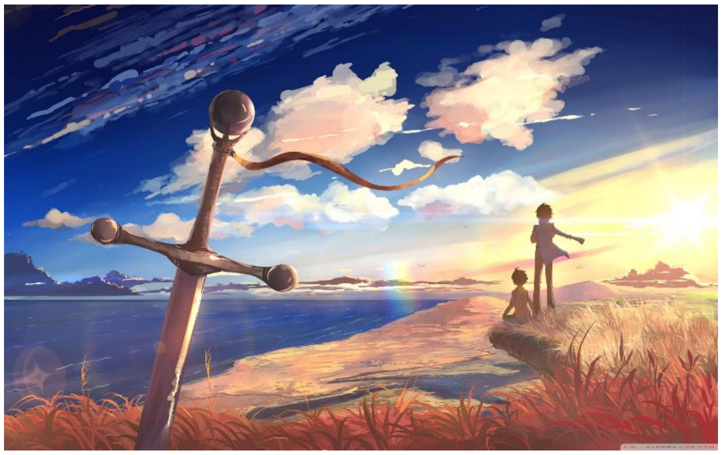


```
void Image::saturate(double amount){

    for(unsigned x = 0; x < width() ; x++)
      for(unsigned y = 0; y < height(); y++)
      {
         //reference on the pixel
         HSLAPixel &P = getPixel(x, y);


         //modifiy the element of P

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

      }
```

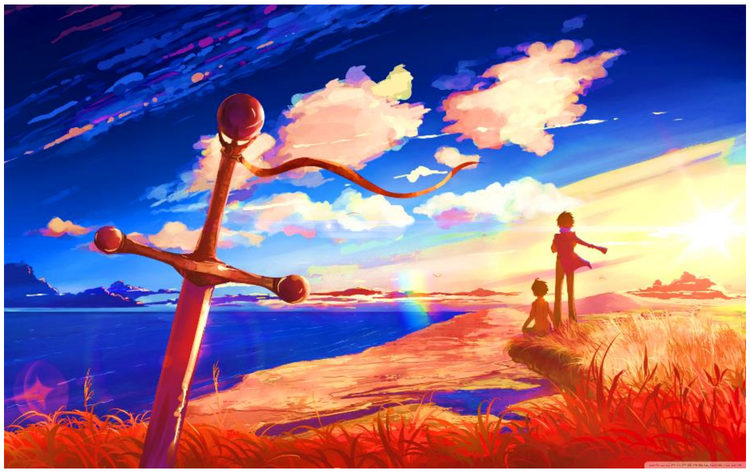
Results

Changing the saturation of the image using saturate(0.5) :

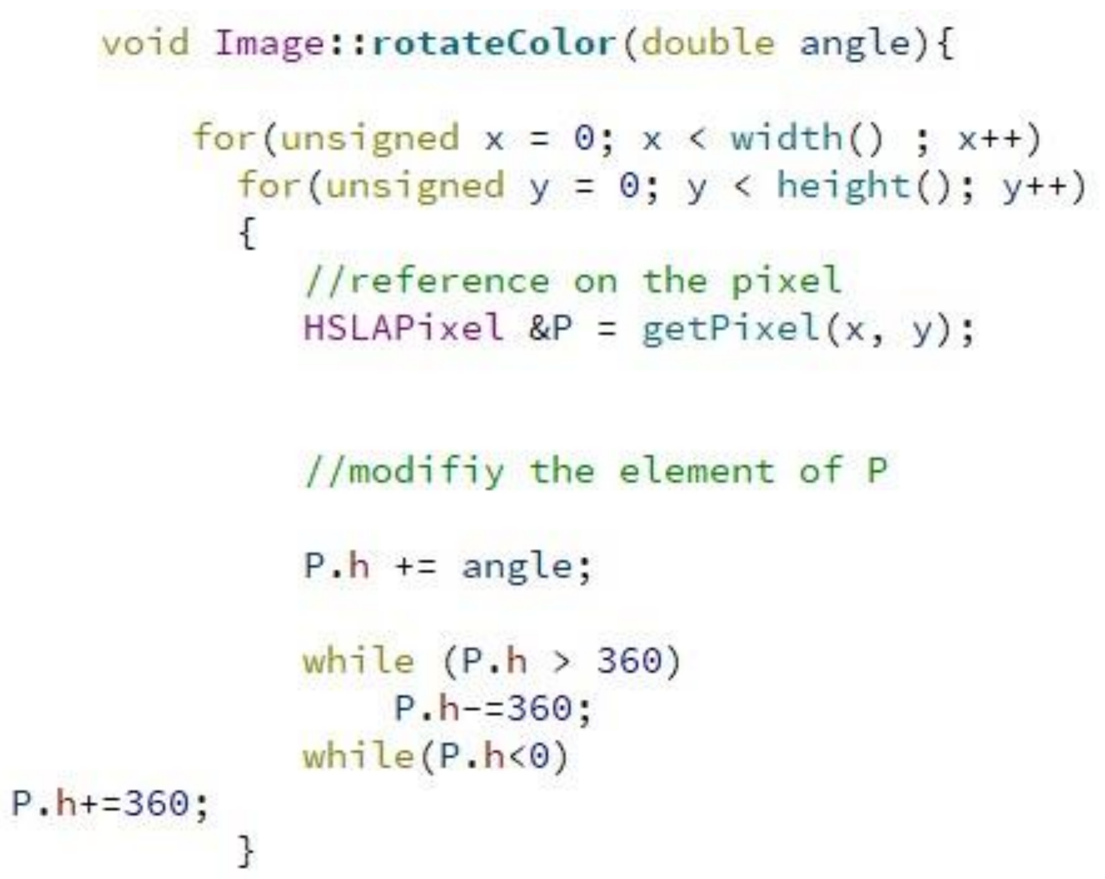
Input:



Output:



- RotateColor(double angle): add the value of angle to each pixel.
    - The value of a color is in cyclic value [0,360].

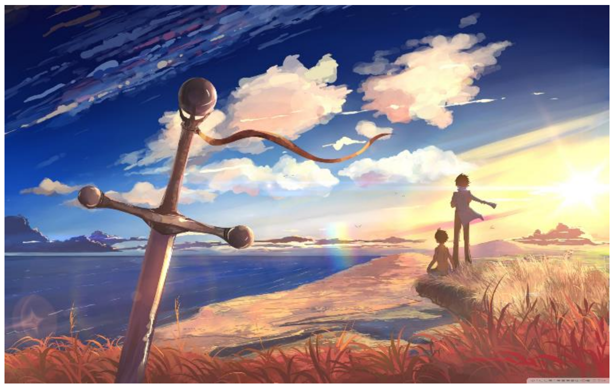


```
void Image::rotateColor(double angle){

    for(unsigned x = 0; x < width() ; x++)
      for(unsigned y = 0; y < height(); y++)
      {
         //reference on the pixel
         HSLAPixel &P = getPixel(x, y);


         //modifiy the element of P

         P.h += angle;

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

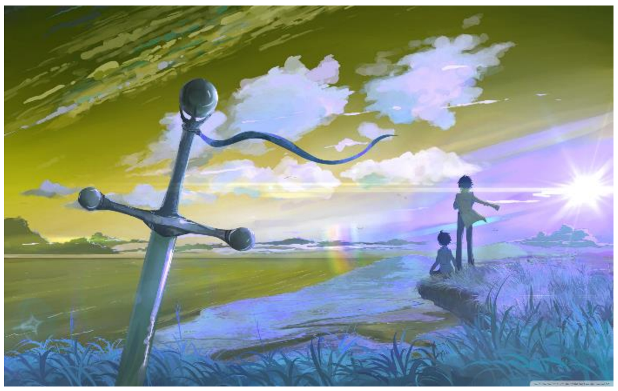
Results

Adding the value of an angle to the image using rotatecolor(200):

Input :

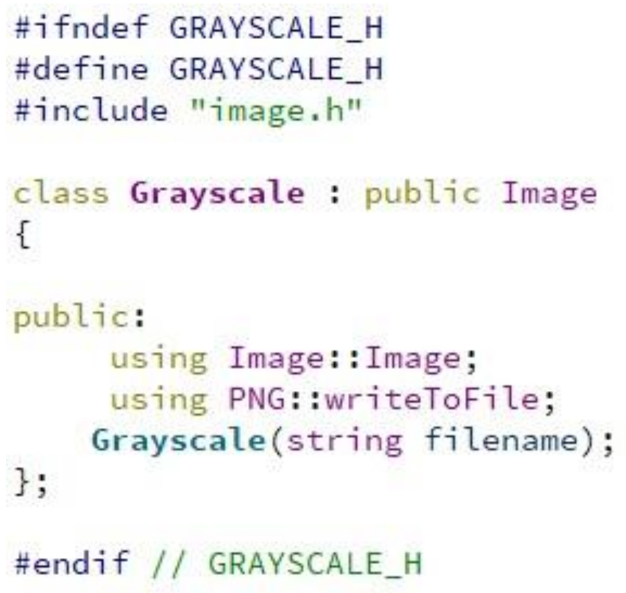


Output :



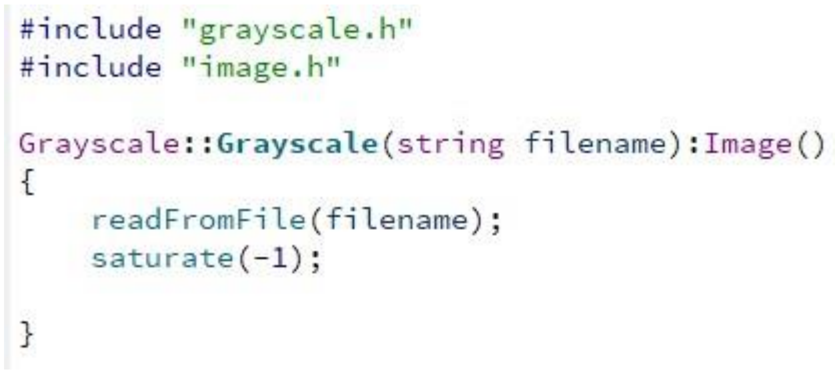
GrayScale class :

Header :



```
#ifndef GRAYSCALE_H
#define GRAYSCALE_H
#include "image.h"

class Grayscale : public Image
{

public:
     using Image::Image;
     using PNG::writeToFile;
    Grayscale(string filename);
};

#endif // GRAYSCALE_H
```

Grayscale.cpp :

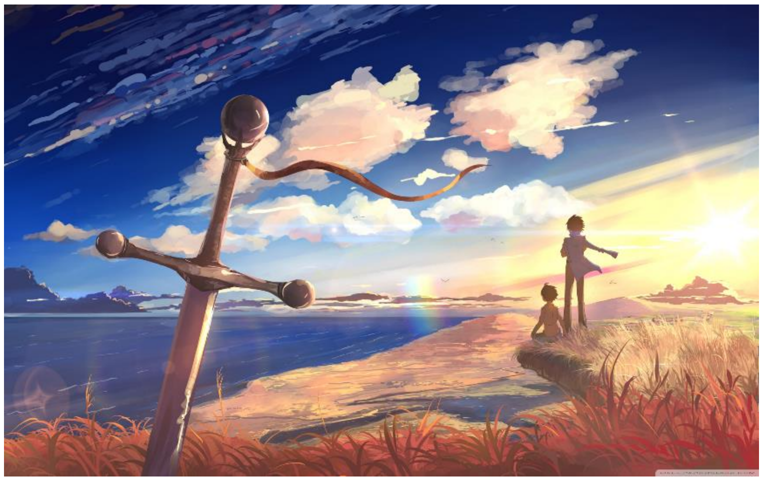


```
#include "grayscale.h"
#include "image.h"

Grayscale::Grayscale(string filename):Image()
{
    readFromFile(filename);
    saturate(-1);

}
```

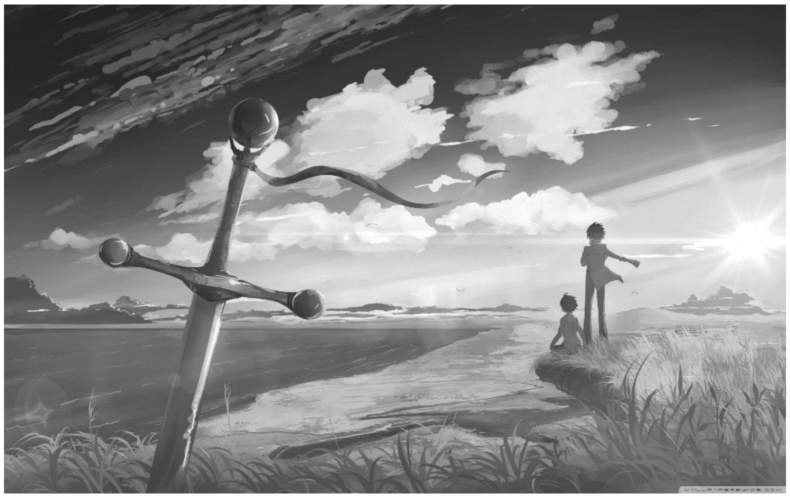
Results

Eliminating all the colors of the image :

Input:

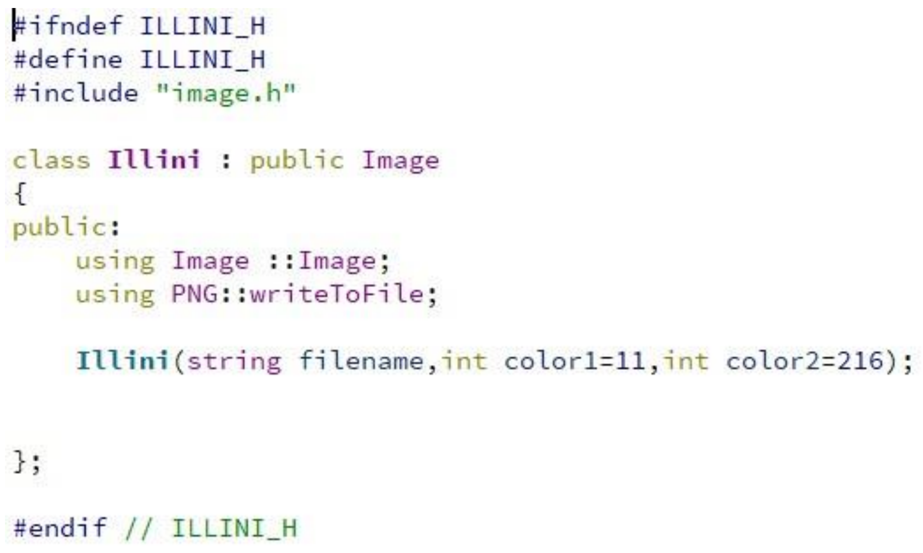


Output:



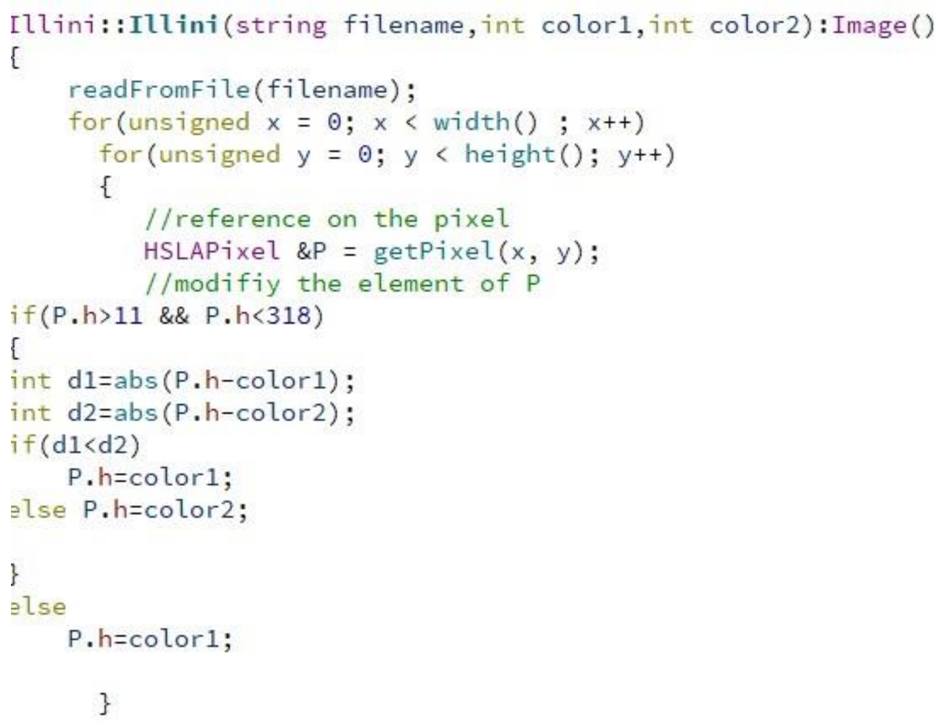
Illini class :

Header :



```
#ifndef ILLINI_H
#define ILLINI_H
#include "image.h"

class Illini : public Image
{
public:
    using Image ::Image;
    using PNG::writeToFile;

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


};

#endif // ILLINI_H
```

Illini.cpp :

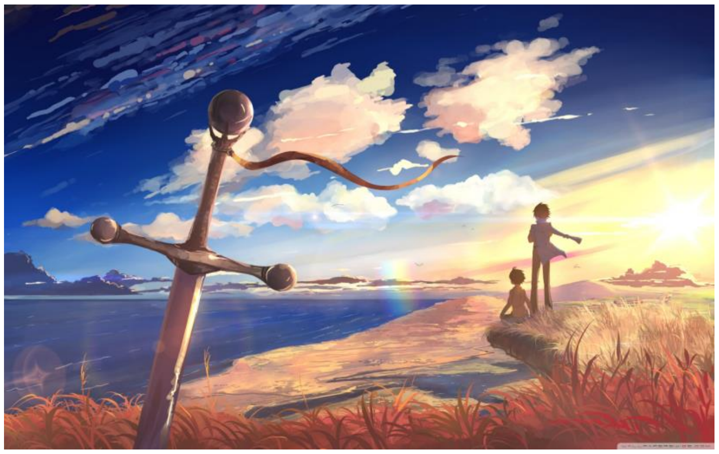


```
Illini::Illini(string filename,int color1,int color2):Image()
{
    readFromFile(filename);
    for(unsigned x = 0; x < width() ; x++)
      for(unsigned y = 0; y < height(); y++)
      {
         //reference on the pixel
         HSLAPixel &P = getPixel(x, y);
         //modifiy the element of P
if(P.h>11 && P.h<318)
{
int d1=abs(P.h-color1);
int d2=abs(P.h-color2);
if(d1<d2)
    P.h=color1;
else P.h=color2;

}
else
    P.h=color1;

      }
```

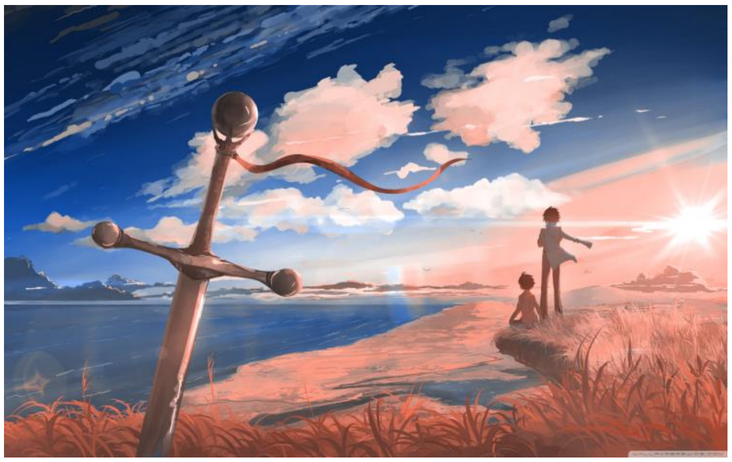
Results

Replacing the hue of each pixel that are either the first or the second color using color1 = 11 (orange) and color2 = 216(blue).

Input :

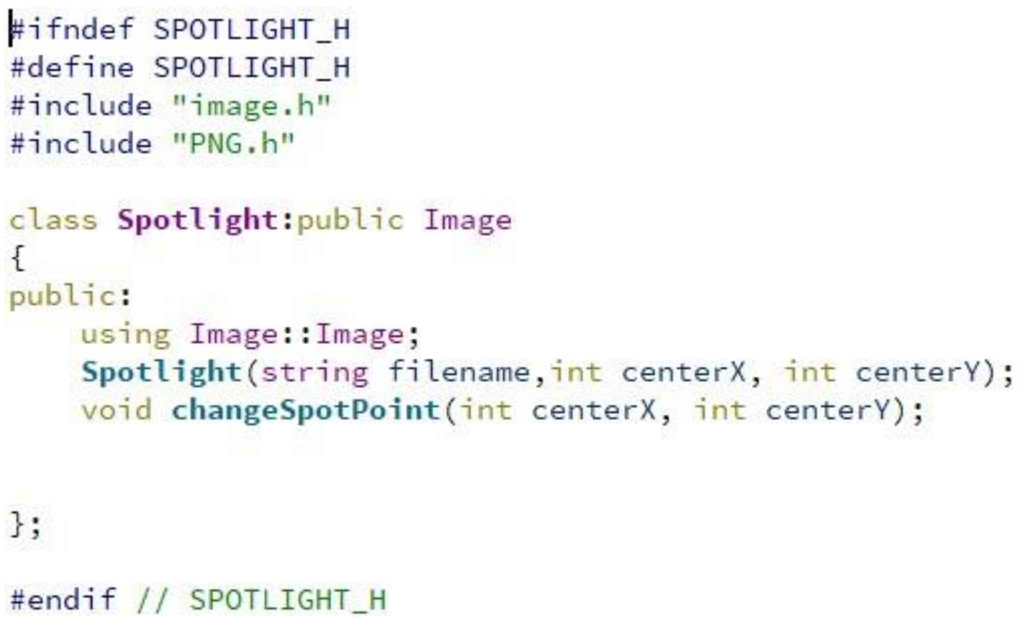


Output :



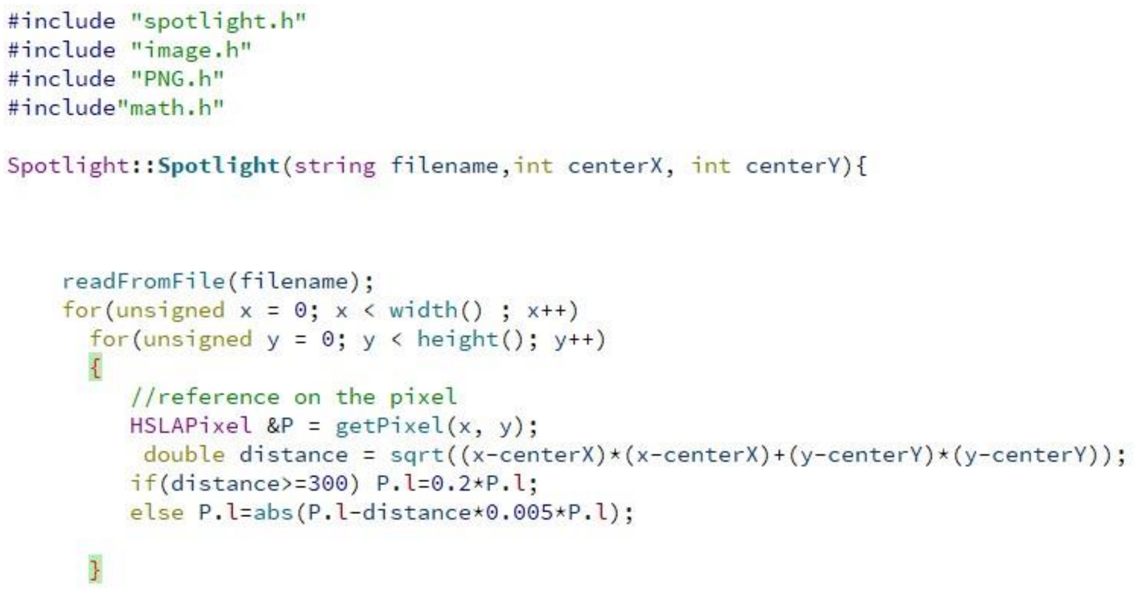
Spotlight class :

Header :



```
#ifndef SPOTLIGHT_H
#define SPOTLIGHT_H
#include "image.h"
#include "PNG.h"

class Spotlight:public Image
{
public:
    using Image::Image;
    Spotlight(string filename,int centerX, int centerY);
    void changeSpotPoint(int centerX, int centerY);


};

#endif // SPOTLIGHT_H
```

Spotlight.cpp:

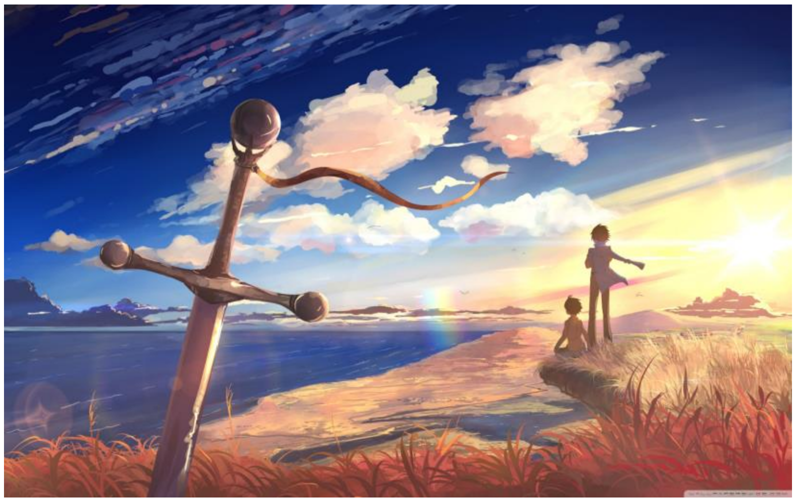


```
#include "spotlight.h"
#include "image.h"
#include "PNG.h"
#include"math.h"

Spotlight::Spotlight(string filename,int centerX, int centerY){


    readFromFile(filename);
    for(unsigned x = 0; x < width() ; x++)
      for(unsigned y = 0; y < height(); y++)
      {
         //reference on the pixel
         HSLAPixel &P = getPixel(x, y);
          double distance = sqrt((x-centerX)*(x-centerX)+(y-centerY)*(y-centerY));
         if(distance>=300) P.l=0.2*P.l;
         else P.l=abs(P.l-distance*0.005*P.l);

      }
```

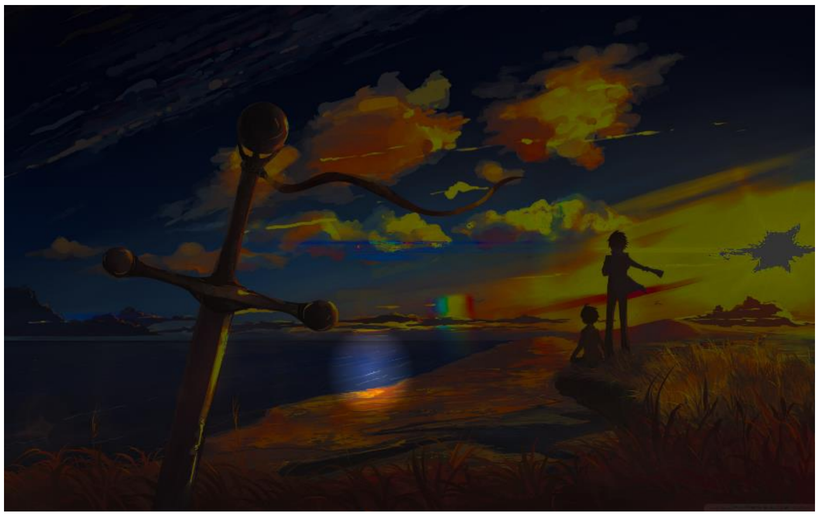
Results

Creating a spotlight centred at the point (1300,1300) :

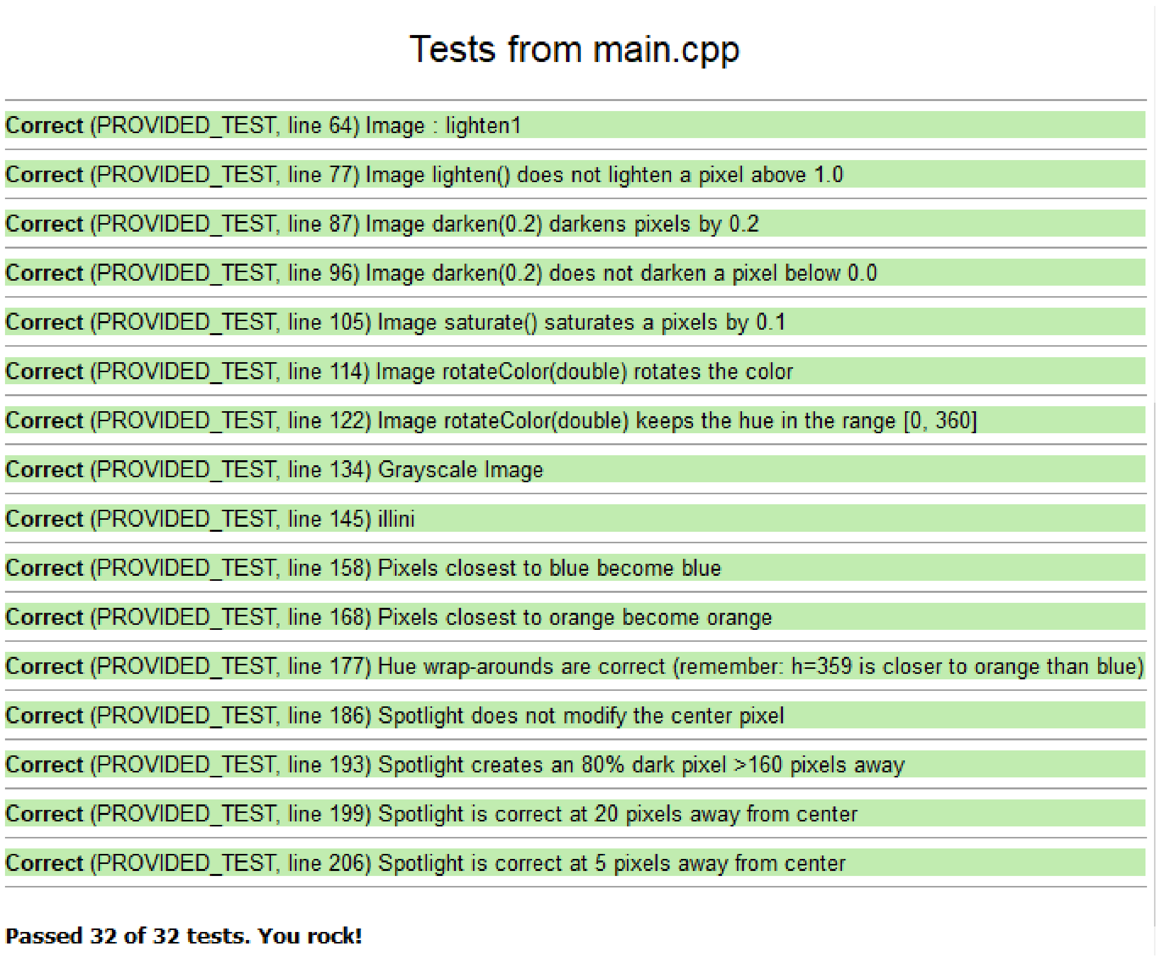
Input:



Output:



Results Of PROVIDED_TESTs :



```
Tests from main.cpp

Correct (PROVIDED_TEST, line 64) Image : lighten1
Correct (PROVIDED_TEST, line 77) Image lighten() does not lighten a pixel above 1.0
Correct (PROVIDED_TEST, line 87) Image darken(0.2) darkens pixels by 0.2
Correct (PROVIDED_TEST, line 96) Image darken(0.2) does not darken a pixel below 0.0
Correct (PROVIDED_TEST, line 105) Image saturate() saturates a pixels by 0.1
Correct (PROVIDED_TEST, line 114) Image rotateColor(double) rotates the color
Correct (PROVIDED_TEST, line 122) Image rotateColor(double) keeps the hue in the range [0, 360]
Correct (PROVIDED_TEST, line 134) Grayscale Image
Correct (PROVIDED_TEST, line 145) illini
Correct (PROVIDED_TEST, line 158) Pixels closest to blue become blue
Correct (PROVIDED_TEST, line 168) Pixels closest to orange become orange
Correct (PROVIDED_TEST, line 177) Hue wrap-arounds are correct (remember: h=359 is closer to orange than blue)
Correct (PROVIDED_TEST, line 186) Spotlight does not modify the center pixel
Correct (PROVIDED_TEST, line 193) Spotlight creates an 80% dark pixel >160 pixels away
Correct (PROVIDED_TEST, line 199) Spotlight is correct at 20 pixels away from center
Correct (PROVIDED_TEST, line 206) Spotlight is correct at 5 pixels away from center

Passed 32 of 32 tests. You rock!
```

thank you
merci
gracias
danke
obrigado
謝謝
go raibh maith agat
teşekkür ederim
dziękuję
bedankt
sukriya
ngiyabonga
спасибо
mochchakkeram
kop khun krap
terima kasih
감사합니다
arigatō
dakujem
мерси
grazie
takk
tapadh leat