```
clear all; close all; im1='Im1.jpg'; im2='Im2.jpg'; im3='Im3.jpg'; im1=imread(im1);
im2=imread(im2); im3=imread(im3); %convertion to double im1=im2double(im1);
im2=im2double(im2); im3=im2double(im3); %normalisation im1=im1/max(max(max(im1)));
im2=im2/max(max(max(im2))); im3=im3/max(max(max(im3))); %display images normalized
%subplot(1,3,1);imshow(im1); %subplot(1,3,2);imshow(im2); %subplot(1,3,3);imshow(im3);
%pause;close; %Exercise 3 %3.1 %The answer is based on the results of Exercise 4 % %It is
better to apply white balancing before nonlinear processing, because %if we apply
nonlinearization before white balancing, the blue light is %more visible, so it falses white and the
other colors. %Exercise 4 %4.1 %mean values of im1 aR=mean(mean(mean(im1(:,:,1))));
aG=mean(mean(mean(im1(:,:,2)))); aB=mean(mean(mean(im1(:,:,3)))); %mean values of im2
bR=mean(mean(mean(im2(:,:,1)))); bG=mean(mean(mean(im2(:,:,2))));
bB=mean(mean(mean(im2(:,:,3)))); %4.2 %average of aR, aG and aB a=(aR+aG+aB)/3;
%average of bR, bG and bB b=(bR+bG+bB)/3; %4.3 %rescaling im1 from aR, aG, aB to a
rescaled_im1(:,:,1)=im1(:,:,1)*(a/aR); rescaled_im1(:,:,2)=im1(:,:,2)*(a/aG);
rescaled_im1(:,:,3)=im1(:,:,3)*(a/aB); %rescaling im2 from bR, bG, bB to b
rescaled_im2(:,:,1)=im2(:,:,1)*(b/bR); rescaled_im2(:,:,2)=im2(:,:,2)*(b/bG);
rescaled_im2(:,:,3)=im2(:,:,3)*(b/bB); %4.4 %Application of non-linear function, then white
balancing. % %convertion to non-linear sRGB sRGB_im1=power(im1(:,:,:), 1/2.4);
sRGB_im2=power(im2(:,:,:), 1/2.4); %Display before white balancing
subplot(1,2,1);imshow(sRGB_im1);title('image 1: convertion to non-linear sRGB before white
balancing'); subplot(1,2,2);imshow(sRGB_im2);title('image 2: convertion to non-linear sRGB
before white balancing'); pause;close; %White balancing %mean values of im1
aR1=mean(mean(mean(sRGB_im1(:,:,1)))); aG1=mean(mean(mean(sRGB_im1(:,:,2))));
aB1=mean(mean(mean(sRGB_im1(:,:,3)))); %mean values of im2
bR1=mean(mean(mean(sRGB_im2(:,:,1)))); bG1=mean(mean(mean(sRGB_im2(:,:,2))));
bB1=mean(mean(mean(sRGB_im2(:,:,3)))); %average of aR, aG and aB a1=(aR1+aG1+aB1)/3;
%average of bR, bG and bB b1=(bR1+bG1+bB1)/3; %rescaling im1 from aR, aG, aB to a
final1_im1(:,:,1)=sRGB_im1(:,:,1)*(a1/aR1); final1_im1(:,:,2)=sRGB_im1(:,:,2)*(a1/aG1);
final1_im1(:,:,3)=sRGB_im1(:,:,3)*(a1/aB1); %rescaling im2 from bR, bG, bB to b
final1_im2(:,:,1)=sRGB_im2(:,:,1)*(b1/bR1); final1_im2(:,:,2)=sRGB_im2(:,:,2)*(b1/bG1);
final1_im2(:,:,3)=sRGB_im2(:,:,3)*(b1/bB1); %display images
subplot(2,2,1);imshow(sRGB_im1);title('image 1: convertion to non-linear sRGB before white
balancing'); subplot(2,2,2);imshow(sRGB_im2);title('image 2: convertion to non-linear sRGB
before white balancing'); subplot(2,2,3);imshow(final1_im1);title('image 1: convertion to
non-linear sRGB then white balancing'); subplot(2,2,4);imshow(final1_im2);title('image 2:
convertion to non-linear sRGB then white balancing'); pause;close; %White balancing, then
application of non-linear function. % %white balancing already done before: we take
rescaled_im1 and rescaled_im2 %non-linearization final2_im1=power(rescaled_im1(:,:,:), 1/2.4);
final2_im2=power(rescaled_im2(:,:,:), 1/2.4); %display images
subplot(2,2,1);imshow(final1_im1);title('image 1: non-linearization, then white balancing');
subplot(2,2,2);imshow(final1_im2);title('image 2: non-linearization, then white balancing');
subplot(2,2,3);imshow(final2_im1);title('image 1: white balancing, then non-linearization');
subplot(2,2,4);imshow(final2_im2);title('image 2: white balancing, then non-linearization');
pause;close; %4.5 %Grey World give good results with im1 because the white is better in the
%first image and the yellow light has been removed. In the second image, we see %a blue light,
which falses the colors apparence, so Grey Balancing doesn't work well for this image.
%Exercise 5 %5.1 Histogram -> imhist() subplot(3,1,1);imhist(im2(:,:,1), 255);title('red channel
im1'); subplot(3,1,2);imhist(im2(:,:,2),255);title('green channel im1');
```

```
subplot(3,1,3);imhist(im2(:,:,3),255);title('blue channel im1'); pause;close; %5.2 and 5.3 %White balancing: [count_im2R, x_im2R]=imhist(im2(:,:,1),255); [count_im2G, x_im2G]=imhist(im2(:,:,2),255); [count_im2B, x_im2B]=imhist(im2(:,:,3),255); sign_hist2R=sign(count_im2R); sign_hist2G=sign(count_im2G); sign_hist2B=sign(count_im2B); sum_sign2R=sum(sign_hist2R); sum_sign2G=sum(sign_hist2G); sum_sign2B=sum(sign_hist2B); %mean value of the 3 channels mean_im2R=(x_im2R'*sign_hist2R)/sum_sign2R; mean_im2G=(x_im2G'*sign_hist2G)/sum_sign2G; mean_im2B=(x_im2B'*sign_hist2B)/sum_sign2B; %total mean value mean_im2=(mean_im2R+mean_im2G+mean_im2B)/3; %rescaling im2 from mean_im2, mean_im2R, mean_im2G and mean_im2B final3_im2(:,:,1)=im2(:,:,1)*(mean_im2/mean_im2R); final3_im2(:,:,2)=im2(:,:,2)*(mean_im2/mean_im2G); final3_im2(:,:,3)=im2(:,:,3)*(mean_im2/mean_im2B); %non-linearization final3_im2=power(final3_im2(:,:,:), 1/2.4); subplot(1,2,1);imshow(final3_im2);title('image 2: Weighted grey world Algorithm'); subplot(1,2,2);imshow(final2_im2);title('image 2: Grey World Algorithm'); pause;close; %5.4 %The advantage of the weighted grey world algorithm over the simple Grey %World Algorithm is that it don't leave a blue light, which false white %color and all other colors. The light is more natural and the color apparences are respected. %Exercise 6 %6.1 %We find the brightest pixel %For Im1 sum_im1=sum(im1,3); [~,max_im1]=max(sum_im1(:)); [row_1, col_1]=ind2sub(size(sum_im1), max_im1); %For Im2 sum_im2=sum(im2,3); [~,max_im2]=max(sum_im2(:)); [row_2, col_2]=ind2sub(size(sum_im2), max_im2); %For Im3 sum_im3=sum(im3,3); [~,max_im3]=max(sum_im3(:)); [row_3, col_3]=ind2sub(size(sum_im3), max_im3); %6.2 %We divide each channel of the normalized image by it's corresponding brighter pixel %For Im1
MaxRGB_im1(:,:,1)=im1(:,:,1)/im1(row_1,col_1, 1);
MaxRGB_im1(:,:,2)=im1(:,:,2)/im1(row_1,col_1, 2);
MaxRGB_im1(:,:,3)=im1(:,:,3)/im1(row_1,col_1, 3); %For Im2
MaxRGB_im2(:,:,1)=im2(:,:,1)/im2(row_2,col_2, 1);
MaxRGB_im2(:,:,2)=im2(:,:,2)/im2(row_2,col_2, 2);
MaxRGB_im2(:,:,3)=im2(:,:,3)/im2(row_2,col_2, 3); %For Im3
MaxRGB_im3(:,:,1)=im3(:,:,1)/im3(row_3,col_3, 1);
MaxRGB_im3(:,:,2)=im3(:,:,2)/im3(row_3,col_3, 2);
MaxRGB_im3(:,:,3)=im3(:,:,3)/im3(row_3,col_3, 3); %6.3 %non linearization
finalMaxRGB_im1=power(MaxRGB_im1(:,:,:), 1/2.4);
finalMaxRGB_im2=power(MaxRGB_im2(:,:,:), 1/2.4);
finalMaxRGB_im3=power(MaxRGB_im3(:,:,:), 1/2.4);
subplot(1,3,1);imshow(finalMaxRGB_im1);title('image 1: MaxRGB Algorithm');
subplot(1,3,2);imshow(finalMaxRGB_im2);title('image 2: MaxRGB Algorithm');
subplot(1,3,3);imshow(finalMaxRGB_im3);title('image 3: MaxRGB Algorithm'); pause;close; %justification %The results are much better on images 1 and 2 because, in image 3, the %differences between the extreme of the differents is much bigger in this %image, so the value of the brighter pixel is not very meaningfull, and this %is due to all the little white pixels of image 3. %In images 1 and 2, the brighter pixel is easier to find because there is %no big differences between the maximal values of each channel. %6.4 %We ask the user to select the white point of the 3 images imshow(im1);title('Select the white point pixel of im1'); [y_im1, x_im1]=ginput(1); imshow(im2);title('Select the white point pixel of im2'); [y_im2, x_im2]=ginput(1); imshow(im3);title('Select the white point pixel of im3'); [y_im3, x_im3]=ginput(1); close; %MaxRGB Algorithm with the selected points %For Im1
```

```
MaxRGB2_im1(:,:,1)=im1(:,:,1)/im1(int16(x_im1),int16(y_im1), 1);
MaxRGB2_im1(:,:,2)=im1(:,:,2)/im1(int16(x_im1),int16(y_im1), 2);
MaxRGB2_im1(:,:,3)=im1(:,:,3)/im1(int16(x_im1),int16(y_im1), 3); %For Im2
MaxRGB2_im2(:,:,1)=im2(:,:,1)/im2(int16(x_im2),int16(y_im2), 1);
MaxRGB2_im2(:,:,2)=im2(:,:,2)/im2(int16(x_im2),int16(y_im2), 2);
MaxRGB2_im2(:,:,3)=im2(:,:,3)/im2(int16(x_im2),int16(y_im2), 3); %For Im3
MaxRGB2_im3(:,:,1)=im3(:,:,1)/im3(int16(x_im3),int16(y_im3), 1);
MaxRGB2_im3(:,:,2)=im3(:,:,2)/im3(int16(x_im3),int16(y_im3), 2);
MaxRGB2_im3(:,:,3)=im3(:,:,3)/im3(int16(x_im3),int16(y_im3), 3); %nonlinearization
finalMaxRGB2_im1=power(MaxRGB2_im1(:,:,:), 1/2.4);
finalMaxRGB2_im2=power(MaxRGB2_im2(:,:,:), 1/2.4);
finalMaxRGB2_im3=power(MaxRGB2_im3(:,:,:), 1/2.4); %Display of the results
subplot(2,3,1);imshow(finalMaxRGB2_im1);title('image 1: MaxRGB Algorithm with white point
manually selected'); subplot(2,3,2);imshow(finalMaxRGB2_im2);title('image 2: MaxRGB
Algorithm with white point manually selected');
subplot(2,3,3);imshow(finalMaxRGB2_im3);title('image 3: MaxRGB Algorithm with white point
manually selected'); subplot(2,3,4);imshow(finalMaxRGB_im1);title('image 1: MaxRGB
Algorithm with automatical selection of the white pixel');
subplot(2,3,5);imshow(finalMaxRGB_im2);title('image 2: MaxRGB Algorithm with automatical
selection of the white pixel'); subplot(2,3,6);imshow(finalMaxRGB_im3);title('image 3: MaxRGB
Algorithm with automatical selection of the white pixel'); pause;close; %Comparison of the 2
methods %We see that the methode that asks the user to enter manually the white %pixel is
more effective, because it works much better with the third %image, that is brighter and
generally much better. %6.5 %application of the median filter on im3
medianFiltered_im3(:,:,1)=medfilt2(im3(:,:,1)); medianFiltered_im3(:,:,2)=medfilt2(im3(:,:,2));
medianFiltered_im3(:,:,3)=medfilt2(im3(:,:,3)); %application of MaxRGB Algorithm %We find the
brightest pixel of the median filtered im3 sum_im3Filtered=sum(medianFiltered_im3,3);
[~,max_im3Filtered]=max(sum_im3Filtered(:)); [filtered_row_3,
filtered_col_3]=ind2sub(size(sum_im3Filtered), max_im3Filtered); %We divide each channel of
im3 median filtered by the value of the %corresponding channel
medianFiltered_MaxRGB_im3(:,:,1)=medianFiltered_im3(:,:,1)/medianFiltered_im3(filtered_row_
3,filtered_col_3, 1);
medianFiltered_MaxRGB_im3(:,:,2)=medianFiltered_im3(:,:,2)/medianFiltered_im3(filtered_row_
3,filtered_col_3, 2);
medianFiltered_MaxRGB_im3(:,:,3)=medianFiltered_im3(:,:,3)/medianFiltered_im3(filtered_row_
3,filtered_col_3, 3); %nonlinearization
finalMedianMaxRGB_im3=power(medianFiltered_MaxRGB_im3(:,:,:), 1/2.4); %display of the
result and comparison with the result of 6.3
subplot(1,2,1);imshow(finalMedianMaxRGB_im3);title('im3: MaxRGB after median filter');
subplot(1,2,2);imshow(finalMaxRGB_im3);title('im3: MaxRGB without a median filter');
pause;close; %6.6 %The advantage is that the median filter deletes the salt and pepper %noise.
%6.7 %The median filter gives good results when it is applicated on an image %with salt and
pepper noise, in a resonable quantitie. %6.8 %If there is no salt and pepper noise, the median
filter will blure the %image a bit. %If there is too much noise, the filter will not be able to recover
the %original image.
```