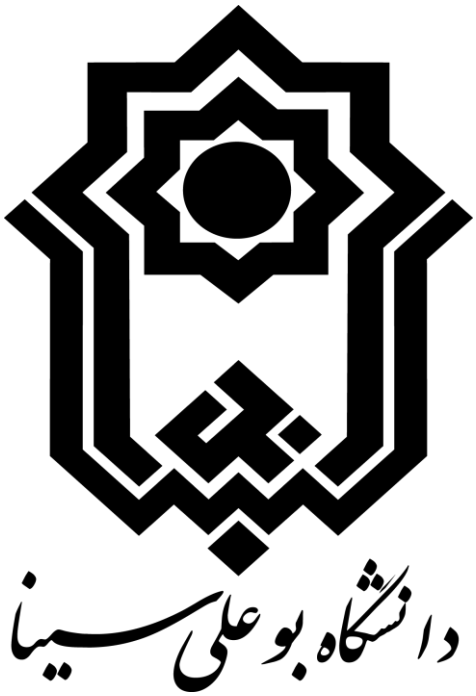
به نام خدا



نام و نام خانوادگی:سید فرهاد حسینی

شماره دانشجویی : ۹۶۱۲۳۵۸۰۱۶

نام درس : مبانی بینایی ماشین

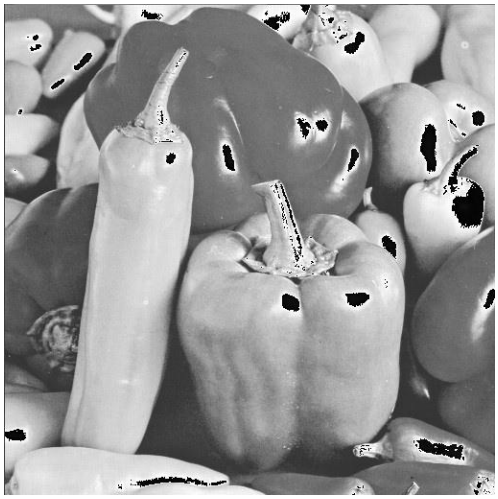
استاد مربوطه : دکترختنلو

• تمرین ۱ :

این تمرین سه بخش دارد در بخش اول ازما خواسته شده که عدد ۵۰ را به تصویر pepper.jpg اضافه کنیم .

تصویر اولیه را در شکل زیر مشاهده میکنید.

وقتی که به تمام پیکسل ها ۵۰ واحد اضافه کنیم تصویر زیر بوجود می آید.



نکته مهم: در کتابخانه opencv در زبان پایتون زمانی که مقدار یک پیکسل از ۲۵۵ تجاوز میکند بطور اتوماتیک مود آن عدد نسبت به ۲۵۶ در آن مکان قرار داده میشود . اما در متلب اینگونه است که اگر یک پیکسل عدد بزرگتر از ۲۵۵ داشته باشد عدد ۲۵۵ بجای آن قرار داده میشود . (به همین دلیل است که تصویر حاصل متفاوت است) .

در بخش دوم هم از ما خواسته شده که پیکسل های بین ۱۲۰ تا ۱۸۰ را به ۵۰ تغییر دهیم .تصویر حاصل بشکل زیر میباشد.

در بخش سوم هم از ما خواسته شده که لگاریتم در مبنای ۱۰ هر پیکسل را محاسبه کنیم .تصویر حاصل بشکل زیر میباشد.

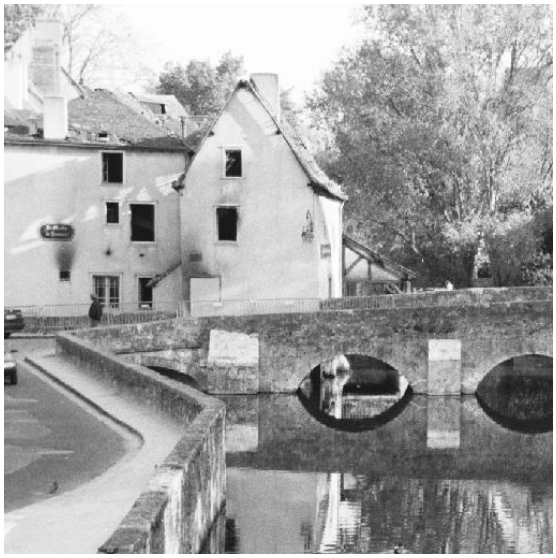
همانطور که انتظار میرفت تصویر حاصل کمی تیره تر از تصویر اولیه شده است .

تمامی کد ها و تصاویر در پوشه Q1 قرار دارند.

• تمرین ۲ :

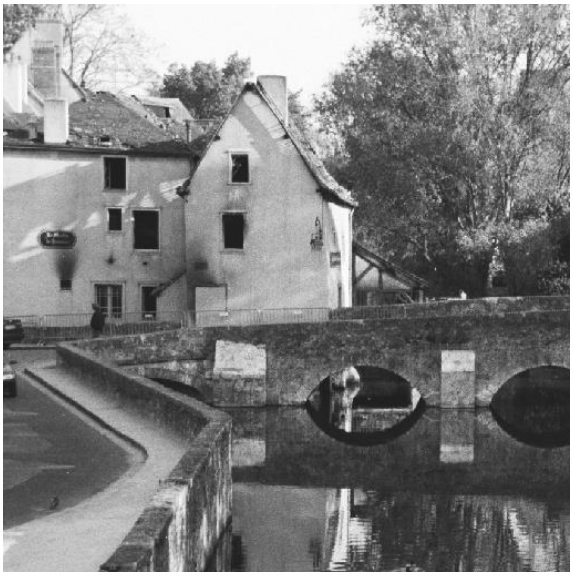
این تمرین سه بخش دارد در بخش اول ازما خواسته شده لگاریتم تصویر در مبنای ۱۰ را حساب کنیم .

ابتدا تصویر اصلی وسپس تصویر لگاریتم را مشاهده میکنیم .(در این سوال مقدار C برابر ۱ در نظر گرفته شده)



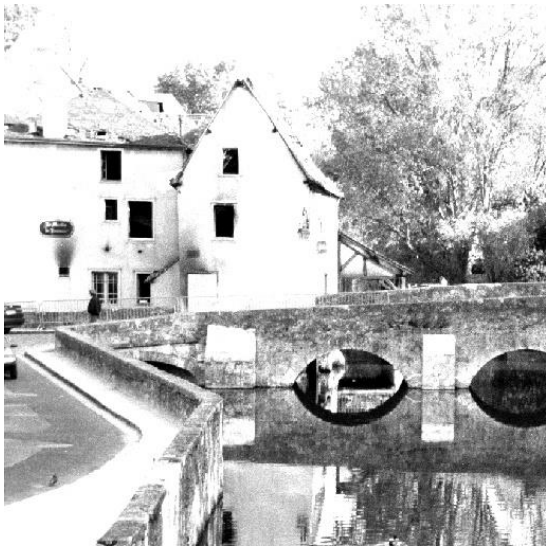
تصویر لگاریتم :

در بخش دوم هم $T(r) = 10^{CR-1}$ این تابع ریاضی را باید برروی نرمال شده تصویر اعمال کنیم .



بنظر می آید که با اعمال این تابع برروی تصویر ، وضوح تصویر کمی بالا رفته و جزئیات بیشتری قابل مشاهده است .

در بخش سوم هم تابع imadjust برروی تصویر اعمال شده . نکته ای که در این بخش قابل ذکر است



این است که تابع آماده imadjust در پایتون وجود نداشت و این تابع طبق فرمول زیر پیاده سازی شد :

```
6   def imadjust(x,a,b,c,d,gamma=1):
7       y = (((x - a) / (b - a)) ** gamma) * (d - c) + c
8       return y
```

علاوه بر تصاویر خواسته شده یکسری اطلاعات آماری هم از ما خواسته شده .(میمنمم ، ماکسیمم ، میانگین ، انحراف ازمعیار):

	مینیمم	ماکسیمم	میانگین	انحراف از معیار
تصویر اصلی	۲۲	۲۵۴	۱۶۱,۱۸۹	۵۸,۰۱
لگاریتم	۲۱,۱۰۲	۱۷۶,۲۵۲	۱۲۲,۳۲۸	۳۶,۷۹
بتوان رسیده	۳۱,۱۰۳	۲۵۲,۷۰۷	۱۲۴,۳۵۲	۶۰,۶۳
adjust	-۱۲,۷۸	۳۵۱,۷۸۵	۲۰۵,۹۴	۹۱,۱۶

در تصویر لگاریتم هر ۴ مقدار آماری نسبت به تصویر اصلی کمتر شده اند . در تصویر هم این موضوع قابل مشاهده بود (تیره تر شدن تصویر) .

در تصویر بتوان رسیده مینیمم کمی زیاد شده و ماکسیمم کمی کمتر شده و اختلاف میانگین و انحراف از معیار هم به نسبت قابل توجی کمتر شده و این بدین معناست که طیف رنگی تصویر فشرده تر شده است (بر خلاف تابع adjust) . کد ها، فایل ها و تصاویر خروجی این تمرین در فولدر Q2 وجود دارد .

• تمرین ۳ :

در این تمرین ابتدا bit plane slicing را با استفاده از یک تابع آماده در متلب پیاده سازی کردیم .

تصویر اصلی :

تمرین اول مبانی بینایی ماشین

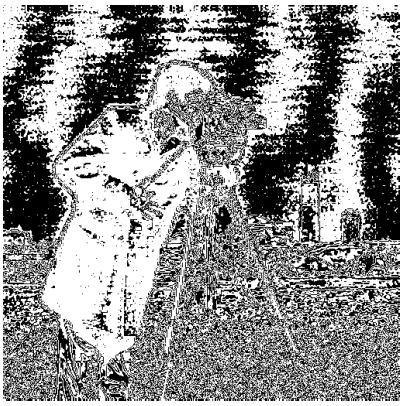
تصویر بیت هشتم (پر ارزش) :

تصویر بیت هفتم:

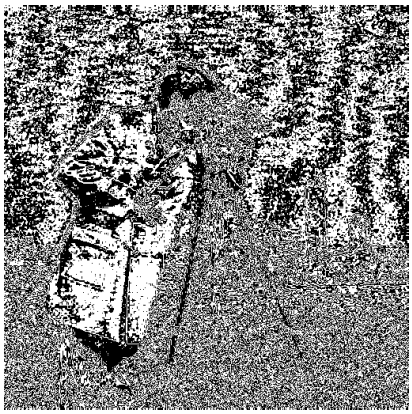
تصویر بیت ششم :

تصویر بیت پنجم :

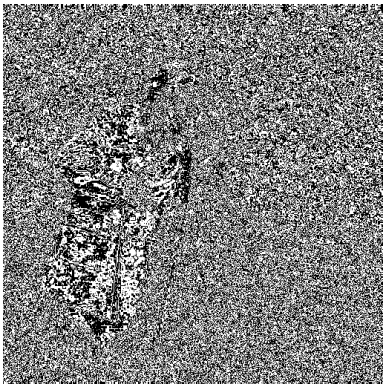
تصویر بیت چهارم :



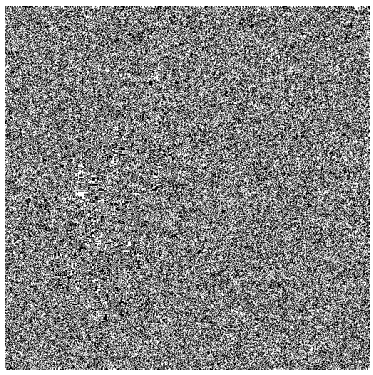
تصویر بیت سوم :



تصویر بیت دوم :



تصویر بیت اول(کم ارزش) :



در بخش استفاده از تابع آماده از تابع bitget() در متلب استفاده شده .

اما در بخش دوم سوال که باید این مکانیزم را بصورت دستی پیاده سازی کنیم روش کار به این صورت است که هربار mod پیکسل هارا نسبت به توانی از دو میگیریم و حاصل را به توانی از دو (یک توان کمتر) تقسیم میکنیم . کد زیر همین موضوع را بیان میکند .

```
B0 = mod(A,2);t=A-B0;
B1 = mod(t,4)/2;t=t-2*B1;
B2 = mod(t,8)/4;t=t-4*B2;
B3 = mod(t,16)/8;t=t-8*B3;
B4 = mod(t,32)/16;t=t-16*B4;
B5 = mod(t,64)/32;t=t-32*B5;
B6 = mod(t,128)/64;t=t-64*B6;
B7 = mod(t,256)/128;t=t-128*B7;
```

در قسمت آخر سوال نیز از ما خواسته شده که با استفاده از ابزار پروفایلر سرعت دو روش بالا را مقایسه کنیم .

روش اول (استفاده از تابع آماده):

Profile Summary

Generated 23-Oct-2020 18:03:09 using performance time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
approach1	1	2.826 s	0.404 s	
imshow	9	2.289 s	0.136 s	
initSize	9	1.902 s	0.028 s	
movegui	9	1.828 s	0.006 s	
legacyMoveGUI	9	1.814 s	1.719 s	
newplot	18	0.139 s	0.038 s	
imread	1	0.119 s	0.006 s	
imread>call_format_specific_reader	1	0.105 s	0.001 s	
imagesci\private\readgif	1	0.103 s	0.002 s	
findall	19	0.101 s	0.092 s	
ima...ate\readgif>read_multiframe_gif	1	0.101 s	0.020 s	
imagesci\private\imgifinfo	1	0.059 s	0.026 s	
basicImageDisplay	9	0.055 s	0.023 s	
newplot>ObserveAxesNextPlot	18	0.055 s	0.006 s	
cla	18	0.049 s	0.006 s	
isSingleImageDefaultPos	9	0.049 s	0.038 s	
graphics\private\clo	19	0.041 s	0.041 s	
newplot>ObserveFigureNextPlot	18	0.028 s	0.002 s	
imageDisplayParseInputs	9	0.026 s	0.002 s	
clf	1	0.026 s	0.006 s	
ima...i\private\initializeMetadataStruct	1	0.026 s	0.003 s	
datestr	1	0.023 s	0.003 s	
imageDisplayValidateParams	9	0.020 s	0.007 s	
timefun\private\dateformverify	1	0.019 s	0.002 s	
...un.imageio.plugins.gif.GIFImageReader (Java method)	4	0.018 s	0.018 s	
timefun\private\formatdate	1	0.017 s	0.010 s	
...;PositionUtils.getDevicePixelPosition	18	0.015 s	0.005 s	

روش دوم (پیاده سازی دستی):

Profile Summary

Generated 23-Oct-2020 18:06:11 using performance time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
approach2	1	2.746 s	0.392 s	
imshow	9	2.231 s	0.122 s	
initSize	9	1.878 s	0.022 s	
movegui	9	1.814 s	0.008 s	
legacyMoveGUI	9	1.797 s	1.703 s	
newplot	18	0.127 s	0.033 s	
imread	1	0.111 s	0.005 s	
findall	19	0.099 s	0.092 s	
imread>call_format_specific_reader	1	0.098 s	0.002 s	
imagesci\private\readgif	1	0.096 s	0.001 s	
ima...ate\readgif>read_multiframe_gif	1	0.094 s	0.020 s	
basicImageDisplay	9	0.051 s	0.022 s	
newplot>ObserveAxesNextPlot	18	0.051 s	0.005 s	
imagesci\private\imgifinfo	1	0.050 s	0.023 s	
isSingleImageDefaultPos	9	0.046 s	0.037 s	
cla	18	0.045 s	0.006 s	
graphics\private\clo	19	0.038 s	0.038 s	
newplot>ObserveFigureNextPlot	18	0.027 s	0.003 s	
imageDisplayParseInputs	9	0.025 s	0.002 s	
clf	1	0.025 s	0.005 s	
ima...i\private\initializeMetadataStruct	1	0.020 s	0.002 s	
...un.imageio.plugins.gif.GIFImageReader (Java method)	4	0.020 s	0.020 s	
imageDisplayValidateParams	9	0.019 s	0.007 s	
datestr	1	0.018 s	0.002 s	
timefun\private\dateformverify	1	0.016 s	0.001 s	

اینطور که بنظر می آید روشی که دستی پیاده سازی شده سریعتر میباشد .

کد ها، فایل ها و تصاویر خروجی این تمرین در فولدر Q3 وجود دارد .

• تمرین ۴ :

در این تمرین ابتدا تمام پیکسل های تصویر اصلی را با ۵۰ جمع میکنیم .(تصویر روشنتر میشود)

تصویر اصلی :

تصویر +۵۰ :

در بخش چهارم این سوال تابع g داده شده است و بایستی آنرا برروی تصویر ورودی اعمال کنیم .

تصویر g بصورت زیر میشود :

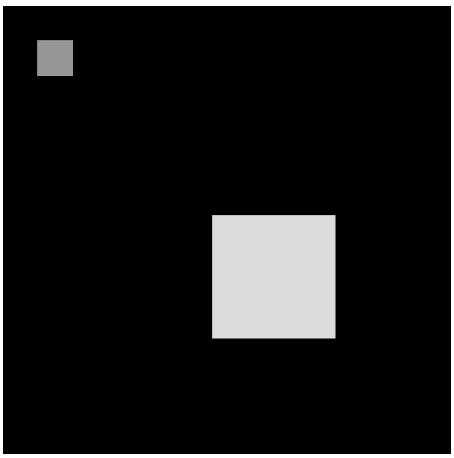
	مینیمم	ماکسیمم	میانگین	میانه	انحراف از معیار	تعداد پیکسل های ۱۲۸
تصویراصلی	۰	۱۹۸	۸۱,۹۵	۸۰	۲۶,۸۱	۱۵۱۸
تصویر +۵۰	۵۰	۲۴۸	۱۳۱,۹۵	۱۳۰	۲۶,۸۱	۲۷۹۵
تصویر g	۰	۲۵۵	۱۰۵,۵۵	۱۰۳,۰۳	۳۴,۵۳	۰

در تصویر دوم (+۵۰) : واضح است که مینیمم و ماکسیمم بایستی با ۵۰ جمع شود . برای مینگین نیز وقتی تمام اعداد که در میانگین مشارکت دارند با ۵۰ جمع شوند میانگین هم با ۵۰ جمع میشود . اما پراکندگی همان پراکندگی قبلی است و انحراف معیار ثابت است .

کد ها، فایل ها و تصاویر خروجی این تمرین در فولدر Q4 وجود دارد .

• تمرین ۵ :

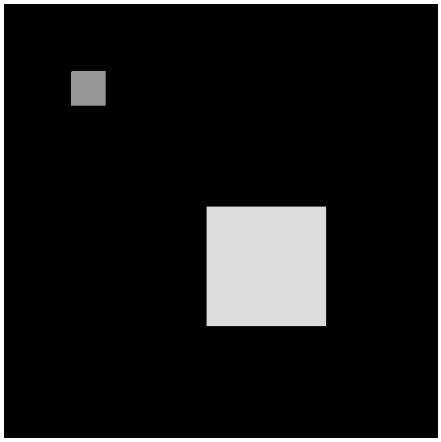
ابتدا تصویر اصلی را مشاهده میکنیم :



در تصصویر بالا مشاهده میشود که دو مربع کوچک و بزرگ داریم . در بخش اول سوال بایستی مربع کوچک را ۴۰ پیکسل در هر بعد جابجا کنیم . این کار توسط حلقه for بسادگی قابل انجام است (لازم بذکر است که ابتدا ما مختصات گوشه بالا سمت چپ مربع و طول ضلع آنرا از قبل بدست آورده ایم . (۳۹,۳۹) و طول ضلع ۴۰).
بصورت زیر :

```
15    for i in range(39,80):
16        for j in range(39,80):
17            img1[i+40][j+40] = img[i][j]
18            img1[i][j] = 0
```

تصویر حاصل بشکل زیر بدست میآید :

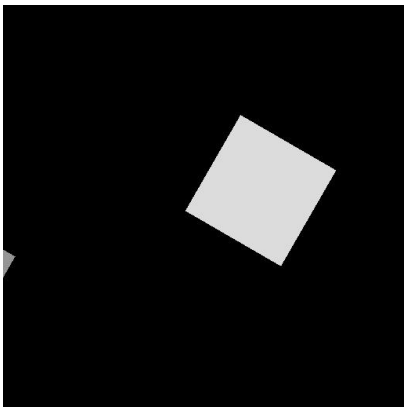


در بخش دوم سوال از ما خواسته شده که مربع بزرگتر را ۶۰ درجه خلاف جهت ساعت بچرخانیم . (برای حل این سوال ، چرخش را حول نقطه ی بالا سمت چپ کل تصویر انجام میدهیم) .

این کار را در دو مرحله انجام میدهیم .

مرحله اول دوران کل عکس حول نقطه (0,0) . این کار بوسیله تابع آماده rotate() انجام میشود .

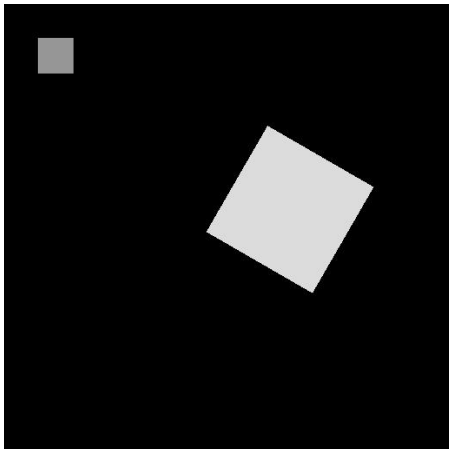
تصویر زیر بدست می آید :



در مرحله دوم این مربع دوران یافته را کپی کرده و بر روی تصویر اصلی پیست میکنیم .(بوسیله دو حلقه for)

```
20    for i in range(90,512):
21        for j in range(100,512):
22            img[i][j] = 0
23            img[i][j] = rot0[i][j]
```

تصویر حاصل بشکل زیر میشود :

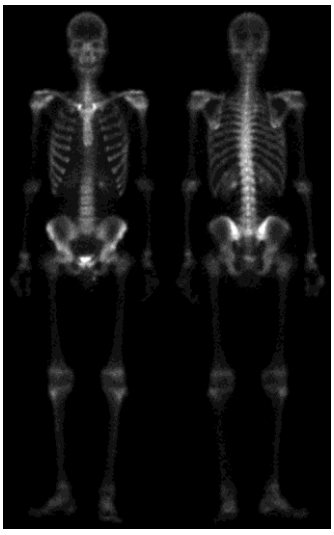


کد ها، فایل ها و تصاویر خروجی این تمرین در فولدر Q5 وجود دارد .

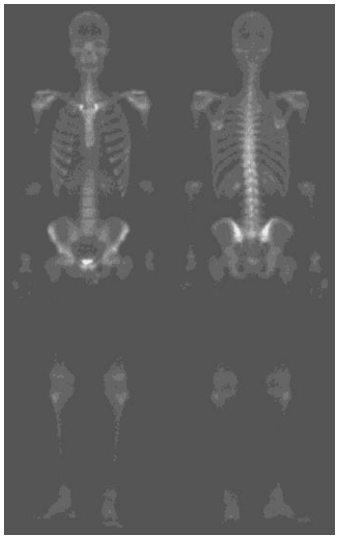
• تمرین ۶ :

در این تمرین نیز همانند تمرین دوم بایستی توابع را برروی تصاویر اعمال کنم . (در این تمرین مقدار c برابر 1/3 در نظر گرفته شده)

تصویر ورودی :



اگر تابع بخش اول را برروی تصویر اعمال کنیم عکس زیر بوجود میآید



اعمال تابع دوم برروی عکس :

اعمال تابع سوم :

تابع سوم بسیار مناسب برای این تصویر میباشد زیرا مناطق حساسی از اسکلت بدن را نمایانتر میکند .

کد ها، فایل ها و تصاویر خروجی این تمرین در فولدر Q6 وجود دارد .

• تمرین ۷ :

در بخش اول این سوال از ما خواسته شده که با مقیاس 0.5 عکس را بزرگنمایی کنیم .

تصویر اصلی :

ضریب 0.5 :

در بخش دوم سوال سه روش داده شده است که طبق آنها بایستی تصاویر را بزرگنمایی کنیم .

در روش نزدیکترین همسایه ، برنامه بصورت دستی نوشته شده است که کد آن بصورت زیر است :

```
for i in range(0 , 512):
    for j in range(0 , 512):
        img1[2*i][2*j] =img[i][j]
        img1[2*i + 1][2*j]=img[i][j]
        img1[2*i][2*j+1] =img[i][j]
        img1[2*i + 1][2*j+1]=img[i][j]
```

تصویر با این روش :

در روش های بعدی صرفا تابع آماده آن که در پایتون وجود دارند فراخوانی شده است :

روش سوم :

کد ها، فایل ها و تصاویر خروجی این تمرین در فولدر Q7 وجود دارد .