سوال اول

برای فیلتر کردن تصویر چهار در چهار با استفاده از فیلتر سه در سه، حاشیه‌ای به طول یک دور تصویر قرار می‌گیرد تا وقتی مرکز فیلتر را روی خانه‌های گوشه‌ی تصویر می‌گذاریم، پنج تا خانه‌ی خالی را پر کنند.
برای کانوالو کردن هم باید فیلتر را 180 درجه بچرخانیم.

Zero padding:

0	0	0	0	0	0
0	1	2	1	6	0
0	7	1	1	1	0
0	3	1	2	0	0
0	1	4	0	2	0
0	0	0	0	0	0

Starting from top left:

1 -> 2 + 1 + 3*7 = 24
2 -> 1 + 1 + 7 + 1*3 + 1 = 13
1 -> 2 + 6 + 1 + 1*3 + 1 = 13
6 -> 1 + 1 + 1*3 = 5
7 -> 1*2 + 2 + 1 + 3*3 + 1 = 15
1 -> 1 + 7 + 3 + 2*2 + 1*3 + 1 + 1 + 2 = 22
1 -> 2 + 1 + 1 + 1*2 + 2*3 + 6 + 1 = 19
1 -> 1 + 1 + 2 + 6*2 = 16
3 -> 7*2 + 1*3 + 1 + 1 + 4 = 23
1 -> 7 + 3 + 1 + 1*2 + 4*3 + 1 + 2= 28
2 -> 1 + 1 + 4 + 2*1 + 1 + 2 = 11
0 -> 1 + 2 + 2*1 + 2*3 = 11
1 -> 3*2 + 1 + 4 = 11
4 -> 3 + 1 + 1*2 + 2 = 8
0 -> 1 + 4 + 2*2 + 2 = 11
2 -> 2

24	13	13	5
15	22	19	16
23	28	11	11
11	8	11	2

Border Reflect:

1	1	2	1	6	6
1	1	2	1	6	6
7	7	1	1	1	1
3	3	1	2	0	0
1	1	4	0	2	2
1	1	4	0	2	2

Starting from top left:

1 -> 24 + 1 + 1 + 7 + 2*1 + 2 = 37
2 -> 13 + 1 + 2*2 + 1 = 19
1 -> 13 + 2 + 1*2 + 6 = 23
6 -> 5 + 6*2 + 6 + 6 + 1 + 1= 31
7 -> 15 + 1 + 7 + 3 = 26
1 -> 22
1 -> 19
1 -> 16 + 6 + 1 = 23
3 -> 23 + 7 + 3 + 1 = 34
1 -> 28
2 -> 11
0 -> 11 + 1 + 2 = 14
1 -> 11 + 3 + 1 + 1 + 1*3 + 4 = 23
4 -> 8 + 1 + 4*3 = 21
0 -> 11 + 4 + 2 = 17
2 -> 2 + 2*3 + 2 + 2 = 12

37	19	23	31
26	22	19	23
34	28	11	14
23	21	17	12

برای چهار پیکسل مرکزی چون در محاسبه نیازی به border ها نداشتیم، تغییری صورت نگرفت. ولی در بقیه‌ی پیکسل‌ها چون حواشی از صفر به مقدار بزرگتر از صفر تبدیل شدند و همه‌ی ضرایب فیلتر مثبتند، باعث افزایش مقدارشان شده است. این افزایش در گوشه‌ها بیشتر بوده چون padding تاثیر بیشتری روی آن‌ها داشته. همچنین تصویر دوم به طور کلی همگن‌تر است.

سوال دوم

الف) کاری که در عملیات ریاضی انجام می‌شود این است که اختلاف شدت نور دو پیکسل همسایه، شدت روشنایی پیکسل اصلی را تعیین می‌کند. در قسمت‌هایی از تصویر که همگن هستند، این اختلاف صفر می‌شود و باعث تیرگی آن قسمت از عکس می‌شود. ولی اگر تغییر شدت روشنایی داشته باشیم، این اختلاف بیشتر می‌شود و پیکسل را روشن‌تر می‌کند. این فیلتر لبه‌های عمودی را تشخیص می‌دهد. چرا که وقتی به پیکسل‌های روی لبه می‌رسد، سمت راست و چپ آن‌ها اختلاف زیادی دارند و به پیکسل اصلی شدت روشنایی زیادی می‌دهند.

ب) این فیلتر از دو فیلتر [0, 1, 0] و [1, 1, 1] تشکیل شده است. یعنی می‌توانیم عکس را یک بار از فیلتر اول رد کنیم و یک بار از فیلتر دوم و نتایج حاصل را با هم جمع کنیم (زیرا برای فیلتر کردن از کانوالو کردن استفاده می‌کنیم و کانلووشن عملیاتی خطی است) فیلتر اول فیلتر همانی است و خروجی آن خود عکس است. فیلتر دوم فیلتر روشن‌کننده و smoothکننده (با این استدلال که اگر ضریب یک سوم داشت، همان فیلتر میانگین گیر بود که از همسایه‌های بالا و پایین برای میانگین گیری استفاده می‌کرد تا نویز را کاهش دهد ولی الان بدون تقسیم بر تعداد کردن فقط جمع می‌کند که باعث روشن‌تر شدن عکس می‌شود) یعنی در آخر ما خود عکس را با خروجی نرمتر و روشنتر شده جمع می‌کنیم.
با توجه به اینکه این فیلتر عمودی است، استفاده‌ی آن برای پدینگ‌های بالا و پایین عکس می‌تواند مفید باشد تا از از بین رفتن لبه‌های جلوگیری کند (تاثیر خود پیکسل را بیشتر کند)

سوال سوم

دو تا پنجره‌ی سه در سه می‌توانیم بگیریم (یکی مرکز 52 و یکی 58).
مرکز 52 برای نیمه‌ی سمت چپی تصویر، clip limit = 1:

n = 9, L = 256

	…	46	51	52	57	58	…
	0	3	1	2	1	2	0
Clipped	1/64	1+1/64	1+1/64	1+1/64	1+1/64	1+1/64	1/64
Cumulative	46/64	1+47/64	2+48/64	3+49/64	4+50/64	5+51/64	…
*= (L-1)/n Rounded	…	49	78	107	135	164	…

مرکز 58 برای نیمه‌ی سمت راستی تصویر، clip limit = 1:

	…	51	52	57	58	59	60	…
	0	1	2	1	2	1	2	0
Clipped	3/256	1 +3/256	1 +3/256	1 +3/256	1 +3/256	1 +3/256	1 +3/256	1 +3/256
Cumulative	153/256	1+156/256	2+159/256	3+162/256	4+165/256	5+168/256	6+171/256	…
*= (L-1)/n Rounded	…	46	74	103	132	160	189	

نتیجه‌ی نهایی این بخش:

49	78	135	160
49	107	132	189
49	107	132	189

مرکز 52 برای نیمه‌ی سمت چپی تصویر، clip limit = 2:

n = 9, L = 256

	…	46	51	52	57	58	…
	0	3	1	2	1	2	0
Clipped	1/256	2+1/256	1+1/256	2+1/256	1+1/256	2+1/256	1/256
Cumulative	46/256	2+47/256	3+48/256	5+49/256	6+50/256	8+51/256	…
*= (L-1)/n Rounded	…	67	90	147	176	232	…

مرکز 58 برای نیمه‌ی سمت راستی تصویر، clip limit = 2:

	…	51	52	57	58	59	60	…
	0	1	2	1	2	1	2	0
Clipped	0	1	2	1	2	1	2	0
Cumulative	0	1	3	4	6	7	9	9
*= (L-1)/n Rounded	…	28	85	113	170	198	255	…

نتیجه‌ی نهایی این بخش:

67	90	113	198
67	147	170	255
67	147	170	255

تفاوت اصلی دستی حساب کردن من و پیاده‌سازی کتابخانه‌ی opencv در دو چیز است: padding و interpolation. پدینگ که واضحا من استفاده نکردم و فقط دو جا پنجره‌های سه در سه را قرار دادم. تابع CLAHE در opencv برای مقادیر مشترک موجود در پنجره ها (خط 229 لینک گیتهاب زیر) و برای به هم وصل کردنشان، از interpolation استفاده شده است که من در دستی اجرا کردن الگوریتم این کار را نکردم و انتخاب کردم که پیکسل های مشترک در هر دو پنجره، از محاسبات کدام پنجره پیروی کنند . در opencv از bilinear interpolation استفاده می‌شود چون ورودی تصویر است و تصویر دوبعدی‌ست. این الگوریتم نسبت به nearest neighbor نتایج smoothتری به ما می‌دهد ولی برای لبه‌ها مناسب نیست.
منابع: github، theailearner، anothertheailearner

سوال چهارم

در پیاده‌سازی این سوال، ذکر موارد زیر ضروری است:
در مواردی که در تصویر مرجع چند مقدار روشنایی متفاوت در متعادل سازی به مقدار یکسانی نگاشته شدند، بزرگترین مقدار روشنایی برای نگاشت انتخاب شده است.
در مواردی که مقدار شدت روشنایی "دقیق" در تصویر مرجع وجود نداشت (مشخص شده با مقدار منفی یک در آرایه‌ی mapping)، مقدار اولین بزرگترین شدت روشنایی برای نگاشت انتخاب شده است.

سوال پنجم

نتایج تابع آماده با پیاده‌سازی خودم تفاوت زیادی نداشت. کیفیت تابع آماده کمی بهتر بود (با اینکه اندازه‌ی کرنل یکی بود) و چند بار ران گرفتم و بیشترین اختلاف بین زمان ها مربوط به خروجی زیر بود. در کل بسیار شبیه بودند.

```
time for applying your guassian filter: 0.001003 s
time for applying OpenCV guassian filter: 0.000801 s
```