check50

cs50/problems/2021/x/filter/more

:) helpers.c exists

Log

```
checking that helpers.c exists...
```

:) filter compiles

Log

```
running clang testing.c helpers.c -o testing -std=c11 -ggdb -lm...
```

:) grayscale correctly filters single pixel with whole number average

Log

```
testing with pixel (20, 40, 90)
running ./testing 0 0...
checking for output "50 50 50\n"...
```

:) grayscale correctly filters single pixel without whole number average

Log

```
testing with pixel (27, 28, 28)
running ./testing 0 1...
checking for output "28 28 28\n"...
```

:) grayscale leaves alone pixels that are already gray

Log

```
testing with pixel (50, 50, 50)
running ./testing 0 2...
checking for output "50 50 50\n"...
```

:) grayscale correctly filters simple 3x3 image

Log

```
testing with sample 3x3 image
first row: (255, 0, 0), (255, 0, 0), (255, 0, 0)
second row: (0, 255, 0), (0, 255, 0), (0, 0, 255)
third row: (0, 0, 255), (0, 0, 255), (0, 0, 255)
running ./testing 0 3...
checking for output "85 85 85\n85 85 85\n85 85 85\n85 85 85\n85 85 85\n85 85 85\n85 85 85\n85 85 85\n85 85 85\n"...
```

:) grayscale correctly filters more complex 3x3 image

Log

```
testing with sample 3x3 image
first row: (10, 20, 30), (40, 50, 60), (70, 80, 90)
second row: (110, 130, 140), (120, 140, 150), (130, 150, 160)
third row: (200, 210, 220), (220, 230, 240), (240, 250, 255)
running ./testing 0 4...
checking for output "20 20 20\n50 50 50\n80 80 80\n127 127 127\n137 137 137\n147 147 147\n210 210 210\n230 230 230\n248 248 248\n"...
```

:) grayscale correctly filters 4x4 image

Log

```
testing with sample 4x4 image
first row: (10, 20, 30), (40, 50, 60), (70, 80, 90), (100, 110, 120)
second row: (110, 130, 140), (120, 140, 150), (130, 150, 160), (140, 160, 170)
third row: (195, 204, 213), (205, 214, 223), (225, 234, 243), (245, 254, 253)
fourth row: (50, 28, 90), (0, 0, 0), (255, 255, 255), (85, 85, 85)
running ./testing 0 5...
checking for output "20 20 20\n50 50 50\n80 80 80\n110 110 110\n127 127 127\n137 137 137\n147 147 147\n157 157 157\n204 204 204\n214 214 214\n234 234 234\n251 251 251\n56 56 56\n0 0 0\n255 255 255\n85 85 85\n"...
```

:) reflect correctly filters 1x2 image

Log

```
testing with sample 1x2 image
first row: (255, 0, 0), (0, 0, 255)
running ./testing 2 0...
checking for output "0 0 255\n255 0 0\n"...
```

:) reflect correctly filters 1x3 image

Log

```
testing with sample 1x3 image
first row: (255, 0, 0), (0, 255, 0), (0, 0, 255)
running ./testing 2 1...
checking for output "0 0 255\n0 255 0\n255 0 0\n"...
```

:) reflect correctly filters image that is its own mirror image

Log

```
testing with sample 3x3 image
first row: (255, 0, 0), (255, 0, 0), (255, 0, 0)
second row: (0, 255, 0), (0, 255, 0), (0, 0, 255)
third row: (0, 0, 255), (0, 0, 255), (0, 0, 255)
running ./testing 2 2...
checking for output "255 0 0\n255 0 0\n255 0 0\n0 255 0\n0 255 0\n0 255 0\n0 0 255\n0 0
255\n0 0 255\n"...
```

:) reflect correctly filters 3x3 image

Log

```
testing with sample 3x3 image
first row: (10, 20, 30), (40, 50, 60), (70, 80, 90)
second row: (110, 130, 140), (120, 140, 150), (130, 150, 160)
third row: (200, 210, 220), (220, 230, 240), (240, 250, 255)
running ./testing 2 3...
checking for output "70 80 90\n40 50 60\n10 20 30\n130 150 160\n120 140 150\n110 130
140\n240 250 255\n220 230 240\n200 210 220\n"...
```

:) reflect correctly filters 4x4 image

Log

```
testing with sample 4x4 image
first row: (10, 20, 30), (40, 50, 60), (70, 80, 90), (100, 110, 120)
second row: (110, 130, 140), (120, 140, 150), (130, 150, 160), (140, 160, 170)
third row: (195, 204, 213), (205, 214, 223), (225, 234, 243), (245, 254, 253)
fourth row: (50, 28, 90), (0, 0, 0), (255, 255, 255), (85, 85, 85)
running ./testing 2 4...
checking for output "100 110 120\n70 80 90\n40 50 60\n10 20 30\n140 160 170\n130 150
160\n120 140 150\n110 130 140\n245 254 253\n225 234 243\n205 214 223\n195 204 213\n85
85 85\n255 255 255\n0 0 0\n50 28 90\n"...
```

:) blur correctly filters middle pixel

Log

```
testing with sample 3x3 image
first row: (10, 20, 30), (40, 50, 60), (70, 80, 90)
second row: (110, 130, 140), (120, 140, 150), (130, 150, 160)
third row: (200, 210, 220), (220, 230, 240), (240, 250, 255)
running ./testing 3 0...
checking for output "127 140 149\n"...
```

:) blur correctly filters pixel on edge

Log

```
testing with sample 3x3 image
first row: (10, 20, 30), (40, 50, 60), (70, 80, 90)
second row: (110, 130, 140), (120, 140, 150), (130, 150, 160)
third row: (200, 210, 220), (220, 230, 240), (240, 250, 255)
running ./testing 3 1...
checking for output "80 95 105\n"...
```

:) blur correctly filters pixel in corner

Log

```
testing with sample 3x3 image
first row: (10, 20, 30), (40, 50, 60), (70, 80, 90)
second row: (110, 130, 140), (120, 140, 150), (130, 150, 160)
third row: (200, 210, 220), (220, 230, 240), (240, 250, 255)
running ./testing 3 2...
checking for output "70 85 95\n"...
```

:) blur correctly filters 3x3 image

Log

```
testing with sample 3x3 image
first row: (10, 20, 30), (40, 50, 60), (70, 80, 90)
second row: (110, 130, 140), (120, 140, 150), (130, 150, 160)
third row: (200, 210, 220), (220, 230, 240), (240, 250, 255)
running ./testing 3 3...
checking for output "70 85 95\n80 95 105\n90 105 115\n117 130 140\n127 140 149\n137 150
159\n163 178 188\n170 185 194\n178 193 201\n"...
```

:) blur correctly filters 4x4 image

Log

```
testing with sample 4x4 image
first row: (10, 20, 30), (40, 50, 60), (70, 80, 90), (100, 110, 120)
second row: (110, 130, 140), (120, 140, 150), (130, 150, 160), (140, 160, 170)
third row: (195, 204, 213), (205, 214, 223), (225, 234, 243), (245, 254, 253)
fourth row: (50, 28, 90), (0, 0, 0), (255, 255, 255), (85, 85, 85)
running ./testing 3 4...
checking for output "70 85 95\n80 95 105\n100 115 125\n110 125 135\n113 126 136\n123
136 145\n142 155 163\n152 165 173\n113 119 136\n143 151 164\n156 166 171\n180 190
194\n113 112 132\n155 156 171\n169 174 177\n203 207 209\n"...
```

:) edges correctly filters middle pixel

Log

```
testing with sample 3x3 image
first row: (0, 10, 25), (0, 10, 30), (40, 60, 80)
second row: (20, 30, 90), (30, 40, 100), (80, 70, 90)
third row: (20, 20, 40), (30, 10, 30), (50, 40, 10)
running ./testing 4 0...
checking for output "210 150 60\n"...
```

:) edges correctly filters pixel on edge

Log

```
testing with sample 3x3 image
first row: (0, 10, 25), (0, 10, 30), (40, 60, 80)
second row: (20, 30, 90), (30, 40, 100), (80, 70, 90)
third row: (20, 20, 40), (30, 10, 30), (50, 40, 10)
running ./testing 4 1...
checking for output "213 228 255\n"...
```

:) edges correctly filters pixel in corner

Log

```
testing with sample 3x3 image
first row: (0, 10, 25), (0, 10, 30), (40, 60, 80)
second row: (20, 30, 90), (30, 40, 100), (80, 70, 90)
third row: (20, 20, 40), (30, 10, 30), (50, 40, 10)
running ./testing 4 2...
checking for output "76 117 255\n"...
```

:) edges correctly filters 3x3 image

Log

```
testing with sample 3x3 image
first row: (0, 10, 25), (0, 10, 30), (40, 60, 80)
second row: (20, 30, 90), (30, 40, 100), (80, 70, 90)
third row: (20, 20, 40), (30, 10, 30), (50, 40, 10)
running ./testing 4 3...
checking for output "76 117 255\n213 228 255\n192 190 255\n114 102 255\n210 150 60\n103
108 255\n114 117 255\n200 197 255\n210 190 255\n"...
```

:) edges correctly filters 4x4 image

Log

```
testing with sample 3x3 image
first row: (0, 10, 25), (0, 10, 30), (40, 60, 80), (50, 60, 80)
second row: (20, 30, 90), (30, 40, 100), (80, 70, 90), (80, 80, 90)
third row: (20, 20, 40), (30, 10, 30), (50, 40, 10), (50, 40, 100)
fourth row: (50, 20, 40), (50, 20, 40), (50, 40, 80), (50, 40, 80)
running ./testing 4 4...
checking for output "76 117 255\n213 228 255\n255 255 255\n255 255 255\n114 102
255\n210 150 60\n177 171 156\n250 247 255\n161 89 255\n126 128 181\n114 170 192\n247
220 192\n148 71 156\n133 100 121\n181 148 212\n212 170 255\n"...
```