

Nome: \_\_\_\_\_ No. USP: \_\_\_\_\_

- A prova está no run.codes e é composta de 4 questões (1, 2, 3 e 4)

- Todas as questões deverão ser feitas num único programa. No início de cada caso de teste existe um inteiro (1,2 3 ou 4) que representa cada questão. Para facilitar seu trabalho, será fornecido um esqueleto de programa (p3.c) cuja função main já contém um switch que invoca cada um das quatro questões. Sugiro fortemente que você o utilize. Com ele, seu trabalho se resume a implementar as 4 questões e funções que achar necessário.

- Não utilize variáveis globais.

Uma imagem pode ser representada em formato ASCII como ilustrada na caixa ao lado. Na primeira linha, há 2 inteiros que indicam, respectivamente, o nro de COLUNAS e LINHAS. Na segunda linha, um inteiro que representa o maior valor do pixel. Na sequência, separados por um espaço em branco (e/ou uma nova linha), os pixels. Uma imagem tem tamanho máximo de 600x400. Os pixels vão de 0 a 255.

```
6 4
255
100 001 020 000 129 100
034 056 006 134 145 200
021 045 078 255 023 210
034 089 234 123 126 190
```

Para representar uma imagem, seu programa deve definir uma estrutura composta por nro de colunas, nro de linhas, valor maximo do pixel e os dados. Os dados devem ser guardados na HEAP (portanto, deve ser alocado dinamicamente). Você pode usar uma representação de matriz OU uma representação linear. Como preferir.

- Defina uma struct **Imagem** com estes elementos.

- Todas as 4 questões terão que ler a imagem. Sugiro fortemente, que você implemente uma função **Imagem** `leImagem()` que lê a imagem dos casos de teste e retorna a imagem lida. Ela poderá ser usada nas quatro questões. Isso evita repetir código desnecessariamente. (1,0 ponto)

- Como os pixels da imagem são armazenados dinamicamente, sugiro que você faça uma função para liberar estes dados sempre que utilizar a função `leImagem()`. (0,5 ponto)

### Questão 1 (1,0 ponto)

A primeira linha indica questão 1. A segunda linha representa um limiar L. O restante é a imagem, já descrita. Seja  $img(i,j)$  o valor do pixel na posição  $i,j$ ,  $0 \leq i \leq 7$  e  $0 \leq j \leq 23$ . Calcule a quantidade de valores tal que  $img(i,j) \geq L$ .

```
1
12
24 7
15
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 3 3 3 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 7 7 7 0 0 11 0 0 0 0 0 15 0 0 15 0
0 3 3 3 0 0 0 7 7 7 7 0 0 11 11 11 0 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 7 7 7 0 0 11 0 0 0 0 0 15 0 0 0 0
0 3 0 0 0 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

A saída esperada para a entrada acima é.

Ha 12 pixels maiores ou iguais a 12

Obs: toda saída, a não ser quando dito o contrário, termina com '\n'.

## Questão 2: (2,5 pontos)

Um histograma é um vetor de ocorrências (quantidade) de todos os pixels da imagens. Na imagem da questão 1, o histograma seria um vetor de dimensão 16 (0 – 15) com o seguinte conteúdo:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
113	0	0	10	0	0	0	20	0	0	0	13	0	0	0	12

Cada entrada do histograma é denominada **bin**. No histograma acima o **bin** tem, portanto, tamanho 1. Se para a imagem do exercício 1 tivéssemos 4 bins (um histograma de dimensão 4) cada entrada do histograma acumularia valores de 4 pixels diferentes, nas faixas de [0..3, 4..7, 8..11, 12..15]. Portanto, o histograma para **bin = 4**, seria:

0	1	2	3
123	20	13	12

A primeira linha indica questão 2. A segunda linha é a quantidade de bins, **qtd**. O restante é a imagem, já descrita. Calcule o histograma de dimensão qtd. Nota: qtd é sempre um valor que divide exatamente o total de pixels da imagem.

2																								
4																								
24	7																							
15																								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	3	3	3	3	0	0	7	7	7	7	0	0	11	11	11	11	0	0	15	15	15	15	0	0
0	3	0	0	0	0	0	7	7	7	7	0	0	11	0	0	0	0	0	15	0	0	15	0	0
0	3	3	3	0	0	0	7	7	7	7	0	0	11	11	11	0	0	0	15	15	15	15	0	0
0	3	0	0	0	0	0	7	7	7	7	0	0	11	0	0	0	0	0	15	0	0	0	0	0
0	3	0	0	0	0	0	7	7	7	7	0	0	11	11	11	11	0	0	15	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Obs: o histograma deve ser alocado dinamicamente e liberado ao final.

A saída esperada é.

```
hist[0] = 123
hist[1] = 20
hist[2] = 13
hist[3] = 12
```

### Questão 3: (2,5 pontos)

Definimos como região de uma imagem, uma coleção de pixels vizinhos de mesmo valor. A Vizinhança pode ser 4-vizinhança ou 8-vizinhança. Na primeira, os pixels vizinhos de  $img(i,j)$  são  $img(i-1,j)$ ,  $img(i+1,j)$ ,  $img(i,j+1)$  e  $img(i,j-1)$ . A 8-vizinhança contém os 4 citados + os pixels nas 4 diagonais. Seu objetivo é escrever uma função que troque os valores de uma região formada por pixels numa 8-vizinhança por um dado valor. É dado um pixel  $img(x,y)$  pertencente à região.

**Você deverá, também, imprimir a quantidade de pixels que foram trocados.**

Nota: a região estará sempre envolta por pixels  $img(i,j) = 0$ . O valor de substituição nunca será zero.

A primeira linha indica questão 3. A segunda linha contém a posição (x,y) de um pixel  $img(x,y)$  pertencente à região e o valor substituto. O restante é a imagem, já descrita.

```
3
1 7 222
24 7
15
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 3 3 3 0 0 7 7 7 7 0 0 11 11 0 11 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 7 7 7 0 0 11 0 11 0 0 0 15 0 0 15 0
0 3 3 3 0 0 0 7 7 7 7 0 0 11 11 11 0 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 7 7 7 0 0 11 0 0 0 0 0 15 0 0 0 0
0 3 0 0 0 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

A saída esperada é:

```
24 7
15
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 3 3 3 0 0 222 222 222 222 0 0 11 11 0 11 0 0 15 15 15 15 0
0 3 0 0 0 0 0 0 222 222 222 222 0 0 11 0 11 0 0 0 15 0 0 15 0
0 3 3 3 0 0 0 0 222 222 222 222 0 0 11 11 11 0 0 0 15 15 15 15 0
0 3 0 0 0 0 0 0 222 222 222 222 0 0 11 0 0 0 0 0 15 0 0 0 0
0 3 0 0 0 0 0 0 222 222 222 222 0 0 11 11 11 11 0 0 15 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Quantidade de pixels trocados = 20
```

Obs: como um pixel pode ser formado por até 3 dígitos (222, por exemplo) a saída de cada pixel deve ser justificada à direita, com espaçamento 4 (para que exista ao menos um espaço em branco entre um pixel e outro). Utilize o conversor "%4d" no printf.

#### Questão 4: (2,5 pontos)

Em uma imagem podemos fazer operações em uma vizinhança. Uma possível operação é criar uma nova imagem onde cada novo pixel (i,j) é a **média** do pixel (i,j) na imagem original + os seus 8 vizinhos imediatos. Seja por exemplo, o pixel  $img(1,1)=3$  da imagem representada no caso de teste abaixo. Podemos criar uma nova imagem (new) e colocar em  $new(1,1)$  a média dada por:

$$new(1,1) = (img(0,0) + img(0,1) + img(0,2) + img(1,0) + img(1,1) + img(1,2) + img(2,0) + img(2,1) + img(2,2)) / 9.$$

Fazendo isso para todo os pixels (i,j), teremos uma nova imagem em que cada pixel (i,j) da nova imagem é a média de  $img(i,j)$  e seus 8 vizinhos. Claramente, o resultado da média será um valor **float**. Sua resposta, portanto, deverá considerar isso. **Imprima, ao final, o valor médio do pixel da nova imagem** (utilize o modificador '%.1f').

Entrada:

A primeira linha indica questão 4. O restante é a imagem, já descrita.

4																							
24	7																						
15																							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	3	3	3	3	0	0	7	7	7	7	0	0	11	11	11	11	0	0	15	15	15	15	0
0	3	0	0	0	0	0	7	7	7	7	0	0	11	0	0	0	0	0	15	0	0	15	0
0	3	3	3	0	0	0	7	7	7	7	0	0	11	11	11	0	0	0	15	15	15	15	0
0	3	0	0	0	0	0	7	7	7	7	0	0	11	0	0	0	0	0	15	0	0	0	0
0	3	0	0	0	0	0	7	7	7	7	0	0	11	11	11	11	0	0	15	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Obs: a média deverá ser calculada para todos os pixels da imagem, exceto os pixels ao longo das 4 bordas (direita, esquerda, superior e inferior). Estes devem permanecer com valor original (no caso destes exemplo, o valor = 0);

A saída esperada é:

24	7																								
15																									
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
0.0	1.0	1.3	1.0	0.7	0.3	1.6	3.1	4.7	4.7	3.1	1.6	2.4	3.7	4.9	3.7	2.4	1.2	3.3	5.0	6.7	6.7	5.0	0.0	0.0	0.0
0.0	1.7	2.3	1.7	1.0	0.3	2.3	4.7	7.0	7.0	4.7	2.3	3.7	6.1	8.6	6.1	3.7	1.2	5.0	8.3	11.7	11.7	8.3	0.0	0.0	0.0
0.0	1.3	1.7	0.7	0.3	0.0	2.3	4.7	7.0	7.0	4.7	2.3	3.7	4.9	6.1	2.4	1.2	0.0	5.0	6.7	8.3	6.7	5.0	0.0	0.0	0.0
0.0	1.3	1.7	0.7	0.3	0.0	2.3	4.7	7.0	7.0	4.7	2.3	3.7	6.1	8.6	6.1	3.7	1.2	5.0	6.7	8.3	5.0	3.3	0.0	0.0	0.0
0.0	0.7	0.7	0.0	0.0	0.0	1.6	3.1	4.7	4.7	3.1	1.6	2.4	3.7	4.9	3.7	2.4	1.2	3.3	3.3	3.3	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Media = 3.6																									

Obs: a saída acima é feita utilizando o modificador "%.1f" no printf (ou seja, cada valor é um float de 'tamanho' 5 (justificado à direita, com 1 casa decimal).

Boa prova!