```c
#include "helpers.h"
#include <math.h>

void verification(int *a);
int se(int line, int column, int height, int width);

// Converte a imagem para escala de cinza(grayscale).
void grayscale(int height, int width, RGBTRIPLE image[height][width])
{
    // Controla a linha que estamos;
    for(int i = 0; i < height; i++)
    {
        // Passa por todas as colunas da imagem;
        for(int j = 0; j < width; j++)
        {
            // Faz a média das 3 cores e arredonda o valor;
            int sum = round(((float)image[i][j].rgbtRed +
image[i][j].rgbtGreen + image[i][j].rgbtBlue) / 3);

            // Iguala todas as cores no valor da média;
            image[i][j].rgbtRed = (image[i][j].rgbtGreen =
(image[i][j].rgbtBlue = sum));
        }
    }
    return;
}

// Converte a imagem para Sépia.
void sepia(int height, int width, RGBTRIPLE image[height][width])
{
    // Controla a linha que estamos;
    for(int i = 0; i < height; i++)
    {
        // Passa por todas as colunas da imagem;
        for(int j = 0; j < width; j++)
        {
            // Faz o cálculo para achar o valor correspondente de cada
cor e arredonda os mesmos;
            int red = round((float)image[i][j].rgbtRed * .393 +
image[i][j].rgbtGreen * .769 + image[i][j].rgbtBlue * .189);
            int green = round((float)image[i][j].rgbtRed * .349  +
image[i][j].rgbtGreen * .686 + image[i][j].rgbtBlue * .168);
            int blue = round((float)image[i][j].rgbtRed * .272 +
image[i][j].rgbtGreen * .534 + image[i][j].rgbtBlue * .131);

            // Verifica se o valor está entre 0x00(0) e 0xff(255);
            verification(&red);
            verification(&green);
            verification(&blue);
```

```
            // Altera as cores deste pixel para os valores encontrados.
            image[i][j].rgbtRed = red;
            image[i][j].rgbtGreen = green;
            image[i][j].rgbtBlue = blue;
        }
    }
    return;
}

// Reflete a imagem horizontalmente.
void reflect(int height, int width, RGBTRIPLE image[height][width])
{
    // Controla a linha em que estamos;
    for(int i = 0; i < height; i++)
    {
        // Passa por metade das colunas da imagem;
        for(int j = 0; j < width / 2; j++)
        {
            // Altera o pixel que estamos com o seu oposto;
            RGBTRIPLE temp = image[i][j];
            image[i][j] = image[i][width - j - 1];
            image[i][width - j - 1] = temp;
        }
    }
    return;
}

// Borra a imagem (Blur).
void blur(int height, int width, RGBTRIPLE image[height][width])
{
    // Variável para armazenar a imagem;
    RGBTRIPLE copy[height][width];

    // Controla a linha que estamos;
    for(int i = 0; i < height; i++)
    {
        // Passa por todas colunas da imagem;
        for(int j = 0; j < width; j++)
        {
            // Copia o pixel da imagem e salva na nova variável.
            copy[i][j] = image[i][j];
        }
    }

    // Controla a linha que estamos;
    for(int i = 0; i< height; i++)
    {
        // Controla a coluna que estamos;
```

```
        for(int j = 0; j < width; j++)
        {
            // Variáveis de controle;
            int sum[3] = {0, 0, 0}, count = 0;

            // Pasa pela linha anterior, atual e posterior ao pixel que
estamos;
            for(int line = i - 1; line < i + 2; line++)
            {
                // Passa pela coluna anterior, atual e posterior ao pixel
que estamos;
                for(int column = j -1; column <  j + 2; column++)
                {
                    // Verifica se o pixel deve ou não ser somado;
                    if(se(line, column, height, width) == 0)
                    {
                        // Soma cada cor em uma parte do array;
                        sum[0] += copy[line][column].rgbtRed;
                        sum[1] += copy[line][column].rgbtGreen;
                        sum[2] += copy[line][column].rgbtBlue;
                        count++;
                    }
                }
            }

            // Tira a média e arredonda a mesma;
            sum[0] = round((float)sum[0] / count);
            sum[1] = round((float)sum[1] / count);
            sum[2] = round((float)sum[2] / count);

            // Verifica se o valor é inferior a 0x00(0) ou superior a
0xff(255);
            verification(&sum[0]);
            verification(&sum[1]);
            verification(&sum[2]);

            // Salva os novos valores na imagem.
            image[i][j].rgbtRed = sum[0];
            image[i][j].rgbtGreen = sum[1];
            image[i][j].rgbtBlue = sum[2];
        }
    }
    return;
}

// Verifica se o valor é inferior a 0x00(0) ou superior a 0xff(255).
void verification(int *a)
{
    if (*a > 0xff)
```

```
    {
        *a = 0xff;
    }
    else if(*a < 0x00)
    {
        *a = 0x00;
    }
}

// Verifica se o pixel verificado deve ou não ser somado;
int se(int line, int column, int height, int width)
{
    // se a linha olhada for menor que 0 ou superior ou igual ao fim da
imagem, impede o mesmo de ser somado;
    if(line < 0 || line >= height)
    {
        return 1;
    }
    // Se a coluna olhada for menor que 0 ou superior ou igual a ultima
colunda da imagem, impede o mesmo de ser somado;
    else if(column < 0 || column >= width)
    {
        return 1;
    }

    // Caso esteja ok, permite esse pixel ser somado.
    return 0;
}
```