```c
#include <getopt.h>
#include <stdio.h>
#include <stdlib.h>

#include "helpers.h"

int main(int argc, char *argv[])
{
    // Ensure proper usage
    if (argc != 3)
    {
        printf("Usage: ./filter infile outfile\n");
        return 1;
    }

    // Remember filenames
    char *infile = argv[1];
    char *outfile = argv[2];

    // Open input file
    FILE *inptr = fopen(infile, "r");
    if (inptr == NULL)
    {
        printf("Could not open %s.\n", infile);
        return 1;
    }

    // Open output file
    FILE *outptr = fopen(outfile, "w");
    if (outptr == NULL)
    {
        fclose(inptr);
        printf("Could not create %s.\n", outfile);
        return 1;
    }

    // Read infile's BITMAPFILEHEADER
    BITMAPFILEHEADER bf;
    fread(&bf, sizeof(BITMAPFILEHEADER), 1, inptr);

    // Read infile's BITMAPINFOHEADER
    BITMAPINFOHEADER bi;
    fread(&bi, sizeof(BITMAPINFOHEADER), 1, inptr);

    // Ensure infile is (likely) a 24-bit uncompressed BMP 4.0
    if (bf.bfType != 0x4d42 || bf.bfOffBits != 54 || bi.biSize != 40 ||
        bi.biBitCount != 24 || bi.biCompression != 0)
    {
        fclose(outptr);
        fclose(inptr);
        printf("Unsupported file format.\n");
        return 1;
    }

    // Get image's height and width
    int height = abs(bi.biHeight);
    int width = bi.biWidth;

    // Allocate memory for image
    RGBTRIPLE(*image)[width] = calloc(height, width * sizeof(RGBTRIPLE));
    if (image == NULL)
    {
```

```c
        printf("Not enough memory to store image.\n");
        fclose(outptr);
        fclose(inptr);
        return 1;
    }

    // Determine padding for scanlines
    int padding = (4 - (width * sizeof(RGBTRIPLE)) % 4) % 4;

    // Iterate over infile's scanlines
    for (int i = 0; i < height; i++)
    {
        // Read row into pixel array
        fread(image[i], sizeof(RGBTRIPLE), width, inptr);

        // Skip over padding
        fseek(inptr, padding, SEEK_CUR);
    }

    // Filter image
    filter(height, width, image);

    // Write outfile's BITMAPFILEHEADER
    fwrite(&bf, sizeof(BITMAPFILEHEADER), 1, outptr);

    // Write outfile's BITMAPINFOHEADER
    fwrite(&bi, sizeof(BITMAPINFOHEADER), 1, outptr);

    // Write new pixels to outfile
    for (int i = 0; i < height; i++)
    {
        // Write row to outfile
        fwrite(image[i], sizeof(RGBTRIPLE), width, outptr);

        // Write padding at end of row
        for (int k = 0; k < padding; k++)
        {
            fputc(0x00, outptr);
        }
    }

    // Free memory
    free(image);

    // Close files
    fclose(inptr);
    fclose(outptr);
    return 0;
}
```