```c
/**
 * copy.c
 *
 * Computer Science 50
 * Problem Set 4
 *
 * Copies a BMP piece by piece, just because.
 */

#include <stdio.h>
#include <stdlib.h>

#include "bmp.h"

int main(int argc, char* argv[])
{
    // ensure proper usage
    if (argc != 3)
    {
        printf("Usage: ./copy 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 2;
    }

    // open output file
    FILE* outptr = fopen(outfile, "w");
    if (outptr == NULL)
    {
        fclose(inptr);
        fprintf(stderr, "Could not create %s.\n", outfile);
        return 3;
    }

    // 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);
        fprintf(stderr, "Unsupported file format.\n");
        return 4;
    }

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

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

    // determine padding for scanlines
    int padding =  (4 - (bi.biWidth * sizeof(RGBTRIPLE)) % 4) % 4;

    // iterate over infile's scanlines
    for (int i = 0, biHeight = abs(bi.biHeight); i < biHeight; i++)
    {
        // iterate over pixels in scanline
        for (int j = 0; j < bi.biWidth; j++)
        {
            // temporary storage
            RGBTRIPLE triple;

            // read RGB triple from infile
            fread(&triple, sizeof(RGBTRIPLE), 1, inptr);

            // change the RGB color structure ***
            if(triple.rgbtRed == 0xFF)
            {
                triple.rgbtGreen = 0xff;
                triple.rgbtBlue = 0xff;
            }
            //TripleRGBTGreen = 000000;
            //TripleRGBTBlue = 000000;

            // write RGB triple to outfile
            fwrite(&triple, sizeof(RGBTRIPLE), 1, outptr);
        }

        // skip over padding, if any
        fseek(inptr, padding, SEEK_CUR);

        // then add it back (to demonstrate how)
        for (int k = 0; k < padding; k++)
        {
            fputc(0x00, outptr);
        }
    }

    // close infile
    fclose(inptr);

    // close outfile
    fclose(outptr);

    // that's all folks
    return 0;
}
```