```c
// Extractor
// Shrub image process and analysis

#include <stdio.h>
#include <stdlib.h>
#include <cs50.h>

#include "bmp.h"

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

    // Open output file
    FILE *outptr = fopen(outfile, "w");
    if (outptr == NULL)
    {
        fclose(inptr);
        printf("extractor: %s: Error creating file\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(inptr);
        fclose(outptr);
        printf("extractor: %s: Unsupported file format\n", infile);
        return 1;
    }

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

```
61
62      // Write outfile's BITMAPINFOHEADER
63      fwrite(&bi, sizeof(BITMAPINFOHEADER), 1, outptr);
64
65      // Iterate over infile's scanlines
66      for (int i = 0, biHeight = abs(bi.biHeight); i < biHeight; i++)
67      {
68          // Iterate over pixels in scanline
69          for (int j = 0; j < bi.biWidth; j++)
70          {
71              // Temporary storage
72              RGBTRIPLE triple;
73
74              // Read RGB triple from infile
75              fread(&triple, sizeof(RGBTRIPLE), 1, inptr);
76
77              // Color match to green shrubs and only allow that through
78              triple.rgbtBlue = 0x00;
79              triple.rgbtRed = 0x00;
80
81              // Write RGB triple to outfile
82              fwrite(&triple, sizeof(RGBTRIPLE), 1, outptr);
83          }
84
85          // Determine scanline's padding
86          int padding = (bi.biWidth * sizeof(RGBTRIPLE)) % 4;
87
88          // Skip over padding, if any
89          fseek(inptr, padding, SEEK_CUR);
90
91          // Write padding to outfile
92          for (int k = 0; k < padding; k++)
93          {
94              fputc(0x00, outptr);
95          }
96      }
97
98      // Close infile
99      fclose(inptr);
100
101     // Close outfile
102     fclose(outptr);
103
104     // Complete
105     return 0;
    }
```