image.h

```
#ifndef IMAGE_H

#define IMAGE_H


typedef struct

{                        // Total: 54 bytes

   uint16_t type;           // Magic identifier: 0x4d42

   uint32_t size;           // File size in bytes

   uint16_t reserved1;       // Not used

   uint16_t reserved2;       // Not used

   uint32_t offset;          // Offset to image data in bytes from beginning of file (54 bytes)

   uint32_t dib_header_size;  // DIB Header size in bytes (40 bytes)

   int32_t width_px;         // Width of the image

   int32_t height_px;        // Height of image

   uint16_t num_planes;      // Number of color planes

   uint16_t bits_per_pixel;   // Bits per pixel

   uint32_t compression;      // Compression type

   uint32_t image_size_bytes; // Image size in bytes

   int32_t x_resolution_ppm;  // Pixels per meter

   int32_t y_resolution_ppm;  // Pixels per meter

   uint32_t num_colors;      // Number of colors

   uint32_t important_colors; // Important colors

} BMPHeader;


typedef struct

{

   BMPHeader header;

   unsigned char header;

   int height;

   int width;
```

```
    int bitDepth;
    unsigned char colorTable;
    unsigned char buffer;
};

int imageReader(const char *, int *, int *, int *, unsigned char *, unsigned char *, unsigned char *);

int imageWriter(const char *, unsigned char *, unsigned char *, unsigned char *, int);

int initialize(const char *, const char *);

#endif
```

image.c

```
// define bmp header size , colorTable and custom image size as per the structure of bmp file.
#define BMP_HEADER_SIZE 54
#define BMP_COLOR_TABLE_SIZE 1024
#define CUSTOM_IMG_SIZE 1024 * 1024

int imageReader(const char *imgName, int *_height, int *_width, int *_bitDepth, unsigned char
*_header, unsigned char *_colorTable, unsigned char *_buffer);
int imageWriter(const char *imgName, unsigned char *header, unsigned char *colorTable, unsigned
char *buffer, int bitDepth);
int initialize(const char *read_image, const char *write_image);

int initialize(const char *read_image, const char *write_image)
{
    // Initialize datatypes of header, height, width, BitDepth to use it after reading.
    int imgWidth, imgHeight, imgBitDepth;
    unsigned char imgHeader[BMP_HEADER_SIZE];
    unsigned char imgColorTable[BMP_COLOR_TABLE_SIZE];
    unsigned char imgBuffer[CUSTOM_IMG_SIZE];


    // Call the read and write function
    int checkReader = imageReader(read_image, &imgWidth, &imgHeight, &imgBitDepth,
&imgHeader[0], &imgColorTable[0], &imgBuffer[0]);
    if (checkReader == 0)
    {
        printf("Read Successful");
    }
    else
    {
        printf("Read Unsuccessful");
    }
    int checkWriter = imageWriter(write_image, imgHeader, imgColorTable, imgBuffer, imgBitDepth);


    if (checkWriter == 0)
    {
        printf("\nWrite Successful");
    }
    else
    {
        printf("\nWrite Unsuccessful");
    }
    return 0;
}
int imageReader(const char *imgName, int *_height, int *_width, int *_bitDepth, unsigned char
*_header, unsigned char *_colorTable, unsigned char *_buffer)
{
    int i;
    // Initialize a FILE pointer for reading
    FILE *streamIn;
    // Open the file to read
```

```
streamIn = fopen(imgName, "rb");
// Check if the FILE pointer is able to access
if (streamIn == (FILE *)0)
{
   printf("Unable to read file\n");
}
for (i = 0; i < 54; i++)
{
   // Read the header.
   _header[i] = getc(streamIn);
}
// Read the width, height, bitDepth from header.
*_width = *(int *)&_header[18];
*_height = *(int *)&_header[22];
*_bitDepth = *(int *)&_header[28];

// Check if the bitDepth is less than 8 and if it is read the colortable from streamIn.
if (*_bitDepth <= 8)
{
   // Read the colortable of size unsigned char from streamIn. 1024 being the
   // no of elements of size mentioned before.
   fread(_colorTable, sizeof(unsigned char), 1024, streamIn);
}
else
{
   printf("BitDepth is greater than 8, can't read the file");
   return 1;
}
// Read the data (buffer) from the streamIn.
fread(_buffer, sizeof(unsigned char), CUSTOM_IMG_SIZE, streamIn);
// Close the FILE pointer.
```

```
    fclose(streamIn);

    return 0;

}

int imageWriter(const char *imgName,

                unsigned char *header,

                unsigned char *colorTable,

                unsigned char *buffer,

                int bitDepth)

{

    // Open the file for write.

    FILE *FO = fopen(imgName, "wb");

    // Write the header of size 54 bytes

    fwrite(header, sizeof(unsigned char), 54, FO);

    // Check to see if the bitDepth is less than 8.

    if (bitDepth <= 8)

    {

        // Write the colortable of size unsigned char from streamIn. 1024 being the

        // no of elements of size mentioned before.

        fwrite(colorTable, sizeof(unsigned char), 1024, FO);

    }

    else

    {

        printf("BitDepth is greater than 8, can't write the file");

        return 1;

    }

    // Write the data.

    fwrite(buffer, sizeof(unsigned char), CUSTOM_IMG_SIZE, FO);

    // Close the write file pointer.

    fclose(FO);

    return 0;

}
```

Main function

main.c

```
#include <stdio.h>

#include <stdlib.h>

#include "image.h"

int main()

{

   // call initialize function and give the arguments of bmp file to read and write.

   initialize("image.bmp", "image_copy.bmp");

}
```

image.bmp

image_copy.bmp