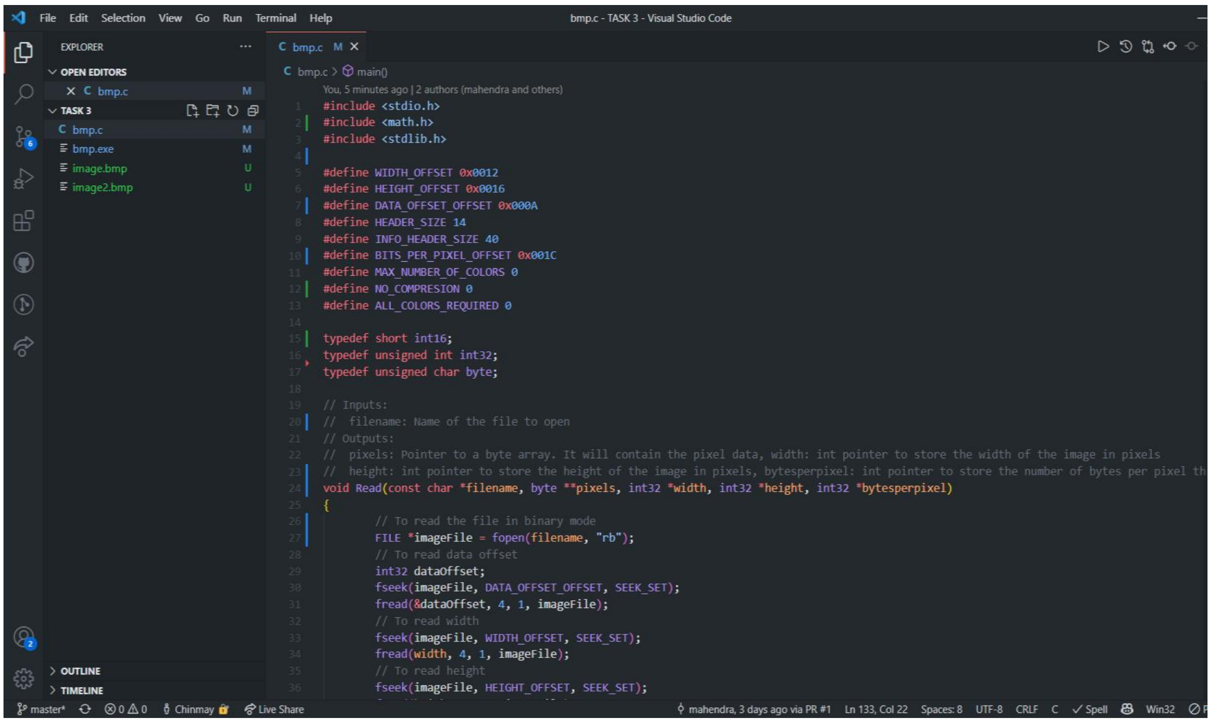
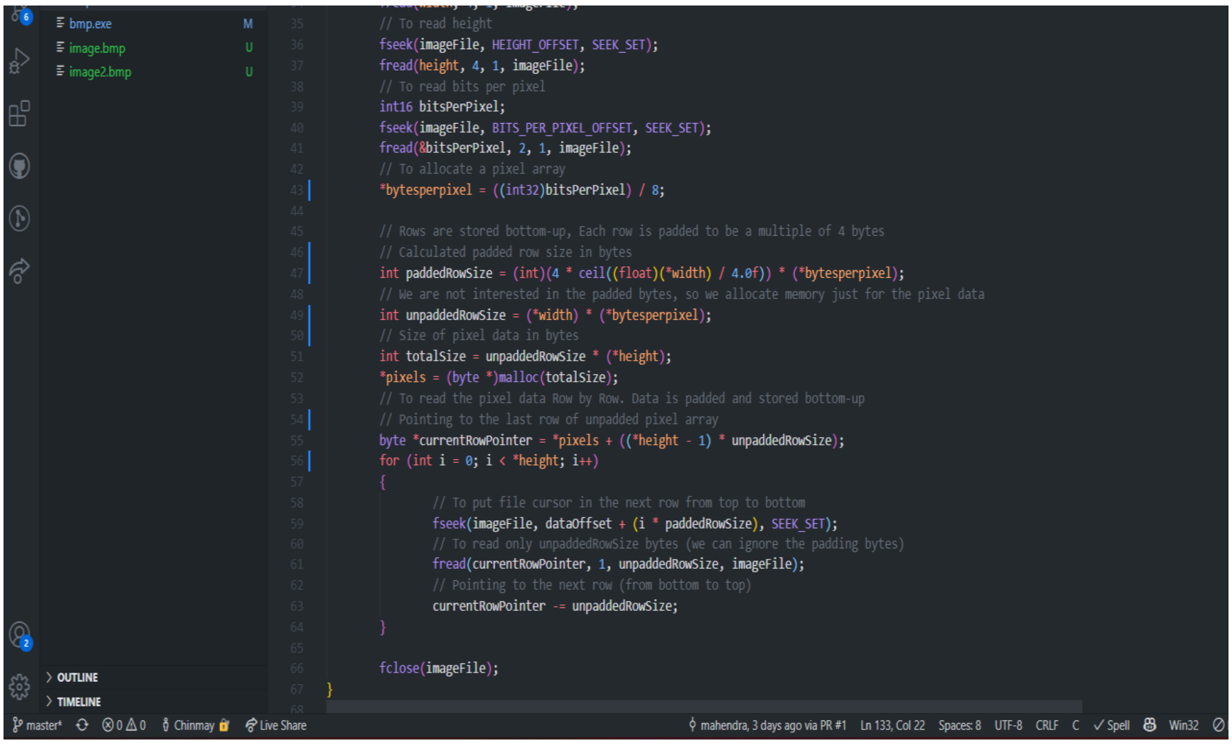
Task #3 Implement an Image File Reader and Writer

1. Function to Read Bitmap Image

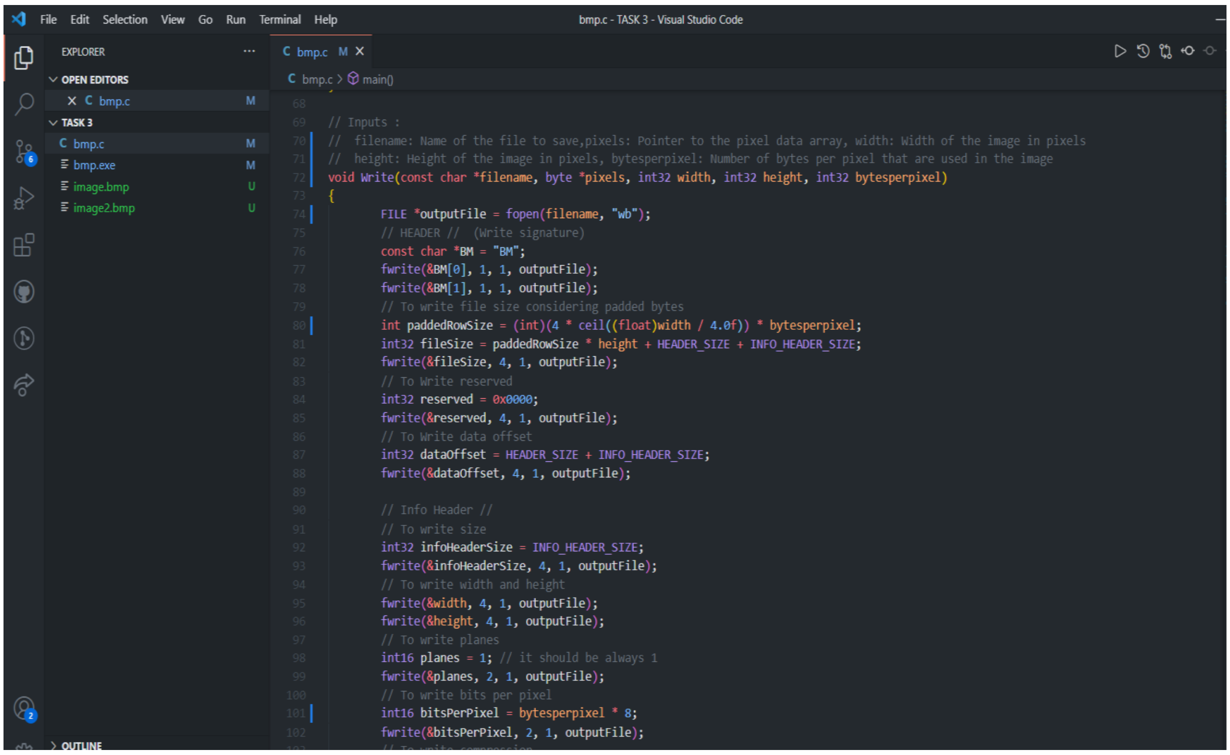


```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define WIDTH_OFFSET 0x0012
#define HEIGHT_OFFSET 0x0016
#define DATA_OFFSET_OFFSET 0x000A
#define HEADER_SIZE 14
#define INFO_HEADER_SIZE 40
#define BITS_PER_PIXEL_OFFSET 0x001C
#define MAX_NUMBER_OF_COLORS 0
#define NO_COMPRESION 0
#define ALL_COLORS_REQUIRED 0

typedef short int16;
typedef unsigned int int32;
typedef unsigned char byte;

// Inputs:
//  filename: Name of the file to open
// Outputs:
//  pixels: Pointer to a byte array. It will contain the pixel data, width: int pointer to store the width of the image in pixels
//  height: int pointer to store the height of the image in pixels, bytesperpixel: int pointer to store the number of bytes per pixel th
void Read(const char *filename, byte **pixels, int32 *width, int32 *height, int32 *bytesperpixel)
{
        // To read the file in binary mode
        FILE *imageFile = fopen(filename, "rb");
        // To read data offset
        int32 dataOffset;
        fseek(imageFile, DATA_OFFSET_OFFSET, SEEK_SET);
        fread(&dataOffset, 4, 1, imageFile);
        // To read width
        fseek(imageFile, WIDTH_OFFSET, SEEK_SET);
        fread(width, 4, 1, imageFile);
        // To read height
        fseek(imageFile, HEIGHT_OFFSET, SEEK_SET);
```

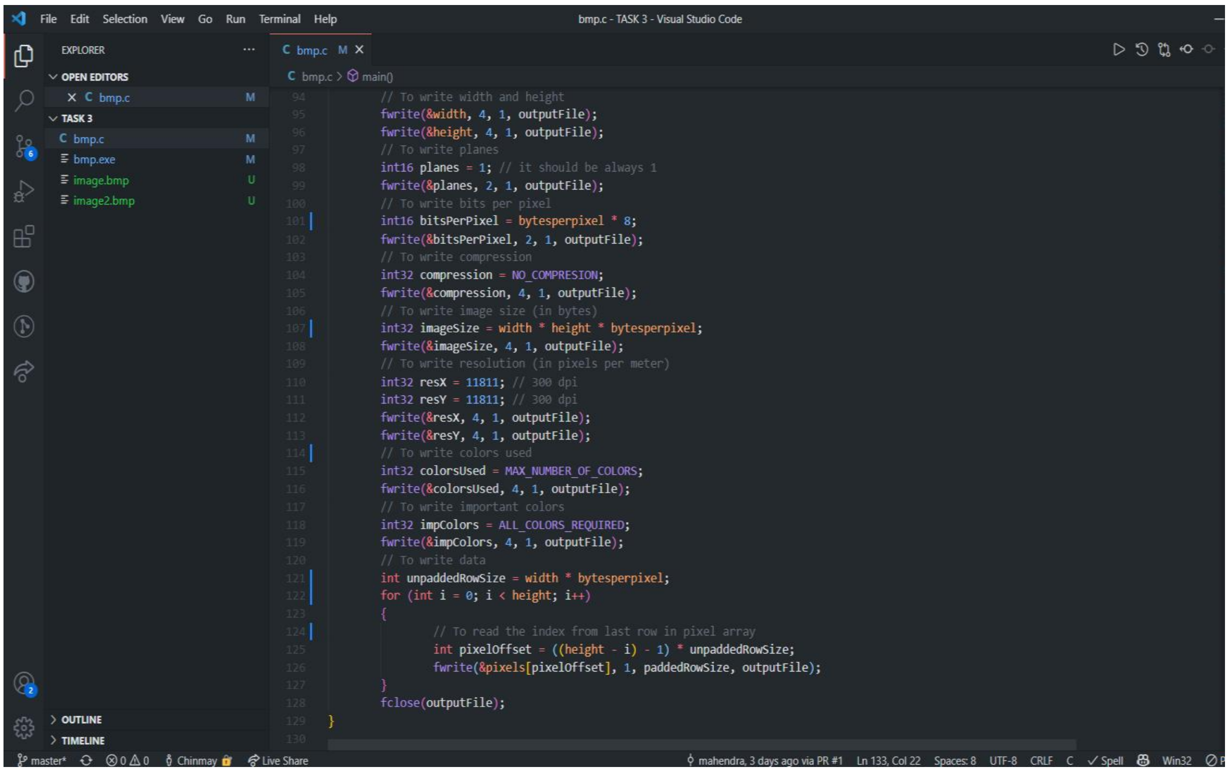


```c
        // To read height
        fseek(imageFile, HEIGHT_OFFSET, SEEK_SET);
        fread(height, 4, 1, imageFile);
        // To read bits per pixel
        int16 bitsPerPixel;
        fseek(imageFile, BITS_PER_PIXEL_OFFSET, SEEK_SET);
        fread(&bitsPerPixel, 2, 1, imageFile);
        // To allocate a pixel array
        *bytesperpixel = ((int32)bitsPerPixel) / 8;

        // Rows are stored bottom-up, Each row is padded to be a multiple of 4 bytes
        // Calculated padded row size in bytes
        int paddedRowSize = (int)(4 * ceil((float)(*width) / 4.0f)) * (*bytesperpixel);
        // We are not interested in the padded bytes, so we allocate memory just for the pixel data
        int unpaddedRowSize = (*width) * (*bytesperpixel);
        // Size of pixel data in bytes
        int totalSize = unpaddedRowSize * (*height);
        *pixels = (byte *)malloc(totalSize);
        // To read the pixel data Row by Row. Data is padded and stored bottom-up
        // Pointing to the last row of unpadded pixel array
        byte *currentRowPointer = *pixels + ((*height - 1) * unpaddedRowSize);
        for (int i = 0; i < *height; i++)
        {
                // To put file cursor in the next row from top to bottom
                fseek(imageFile, dataOffset + (i * paddedRowSize), SEEK_SET);
                // To read only unpaddedRowSize bytes (we can ignore the padding bytes)
                fread(currentRowPointer, 1, unpaddedRowSize, imageFile);
                // Pointing to the next row (from bottom to top)
                currentRowPointer -= unpaddedRowSize;
        }

        fclose(imageFile);
}
```

2. Function to Write Bitmap Image

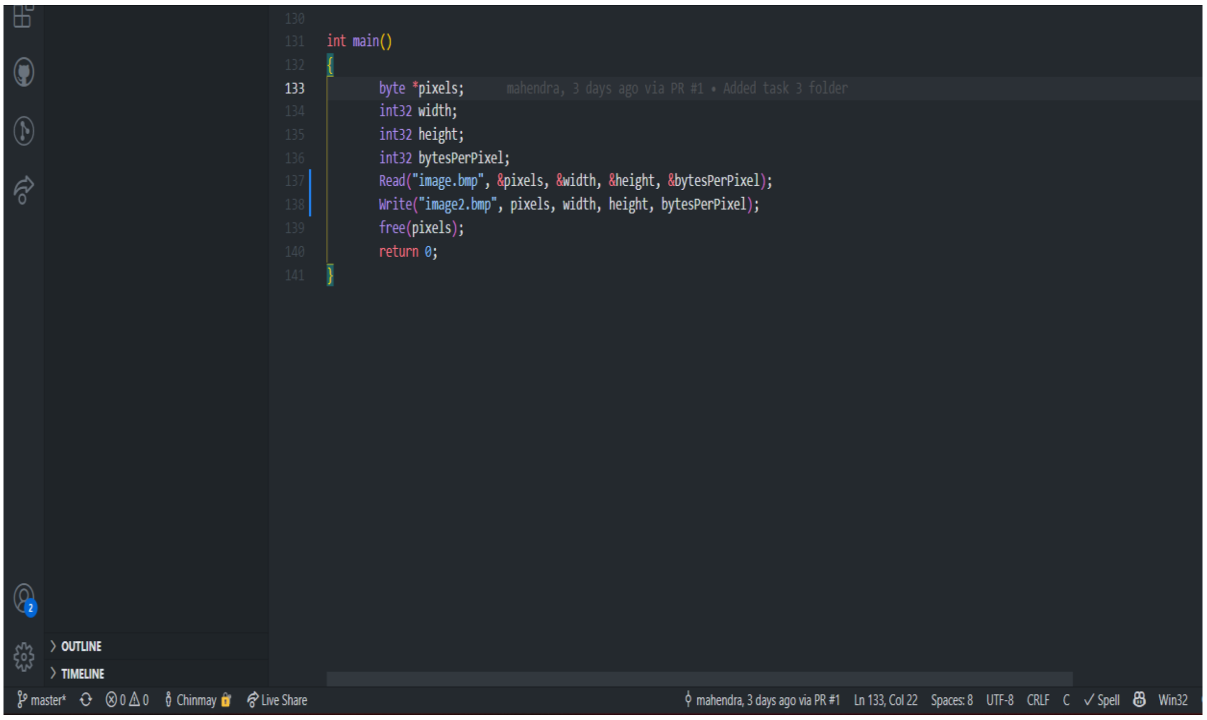


```c

// Inputs :
//  filename: Name of the file to save,pixels: Pointer to the pixel data array, width: Width of the image in pixels
//  height: Height of the image in pixels, bytesperpixel: Number of bytes per pixel that are used in the image
void Write(const char *filename, byte *pixels, int32 width, int32 height, int32 bytesperpixel)
{
        FILE *outputFile = fopen(filename, "wb");
        // HEADER //  (Write signature)
        const char *BM = "BM";
        fwrite(&BM[0], 1, 1, outputFile);
        fwrite(&BM[1], 1, 1, outputFile);
        // To write file size considering padded bytes
        int paddedRowSize = (int)(4 * ceil((float)width / 4.0f)) * bytesperpixel;
        int32 fileSize = paddedRowSize * height + HEADER_SIZE + INFO_HEADER_SIZE;
        fwrite(&fileSize, 4, 1, outputFile);
        // To Write reserved
        int32 reserved = 0x0000;
        fwrite(&reserved, 4, 1, outputFile);
        // To Write data offset
        int32 dataOffset = HEADER_SIZE + INFO_HEADER_SIZE;
        fwrite(&dataOffset, 4, 1, outputFile);

        // Info Header //
        // To write size
        int32 infoHeaderSize = INFO_HEADER_SIZE;
        fwrite(&infoHeaderSize, 4, 1, outputFile);
        // To write width and height
        fwrite(&width, 4, 1, outputFile);
        fwrite(&height, 4, 1, outputFile);
        // To write planes
        int16 planes = 1; // it should be always 1
        fwrite(&planes, 2, 1, outputFile);
        // To write bits per pixel
        int16 bitsPerPixel = bytesperpixel * 8;
        fwrite(&bitsPerPixel, 2, 1, outputFile);
```


File Edit Selection View Go Run Terminal Help
bmp.c - TASK 3 - Visual Studio Code
EXPLORER
OPEN EDITORS
bmp.c M
TASK 3
bmp.c M
bmp.exe M
image.bmp U
image2.bmp U
OUTLINE
TIMELINE
bmp.c > main()
master* 0 0 Chinmay Live Share
mahendra, 3 days ago via PR #1 Ln 133, Col 22 Spaces: 8 UTF-8 CRLF C Spell Win32

```c
        // To write width and height
        fwrite(&width, 4, 1, outputFile);
        fwrite(&height, 4, 1, outputFile);
        // To write planes
        int16 planes = 1; // it should be always 1
        fwrite(&planes, 2, 1, outputFile);
        // To write bits per pixel
        int16 bitsPerPixel = bytesperpixel * 8;
        fwrite(&bitsPerPixel, 2, 1, outputFile);
        // To write compression
        int32 compression = NO_COMPRESION;
        fwrite(&compression, 4, 1, outputFile);
        // To write image size (in bytes)
        int32 imageSize = width * height * bytesperpixel;
        fwrite(&imageSize, 4, 1, outputFile);
        // To write resolution (in pixels per meter)
        int32 resX = 11811; // 300 dpi
        int32 resY = 11811; // 300 dpi
        fwrite(&resX, 4, 1, outputFile);
        fwrite(&resY, 4, 1, outputFile);
        // To write colors used
        int32 colorsUsed = MAX_NUMBER_OF_COLORS;
        fwrite(&colorsUsed, 4, 1, outputFile);
        // To write important colors
        int32 impColors = ALL_COLORS_REQUIRED;
        fwrite(&impColors, 4, 1, outputFile);
        // To write data
        int unpaddedRowSize = width * bytesperpixel;
        for (int i = 0; i < height; i++)
        {
                // To read the index from last row in pixel array
                int pixelOffset = ((height - i) - 1) * unpaddedRowSize;
                fwrite(&pixels[pixelOffset], 1, paddedRowSize, outputFile);
        }
        fclose(outputFile);
}
```

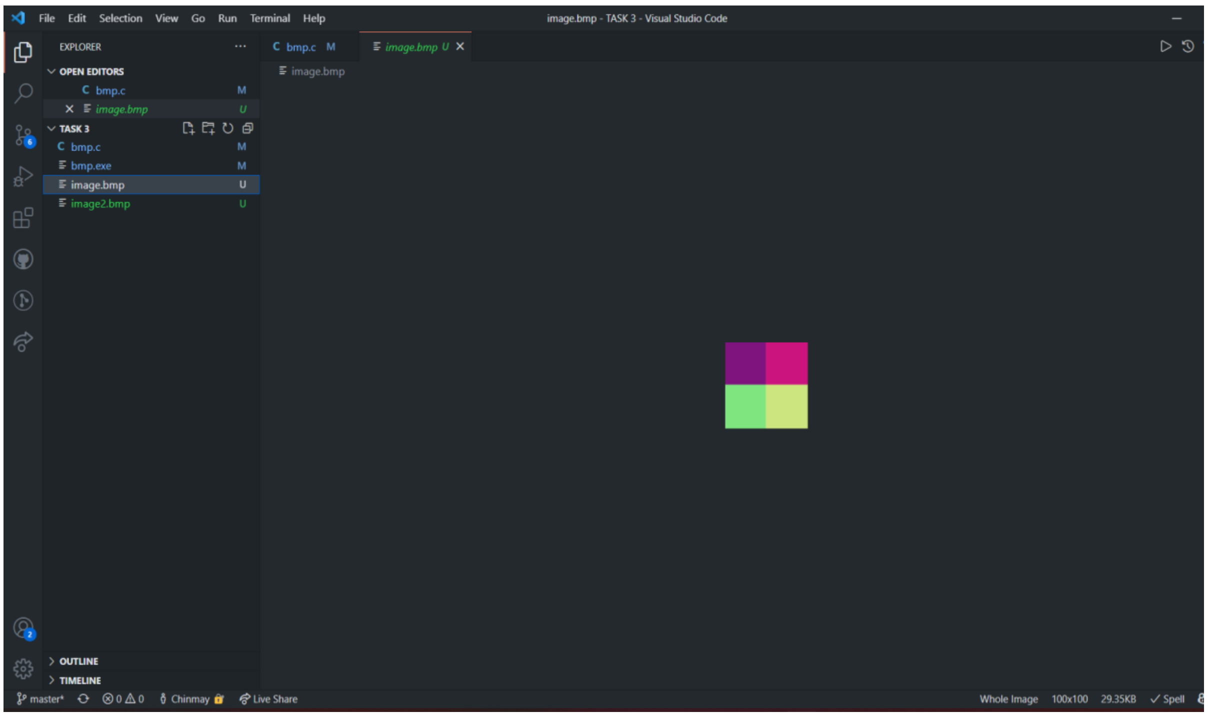
3. Main Function



```
int main()
{
        byte *pixels;      mahendra, 3 days ago via PR #1 • Added task 3 folder
        int32 width;
        int32 height;
        int32 bytesPerPixel;
        Read("image.bmp", &pixels, &width, &height, &bytesPerPixel);
        Write("image2.bmp", pixels, width, height, bytesPerPixel);
        free(pixels);
        return 0;
}
```

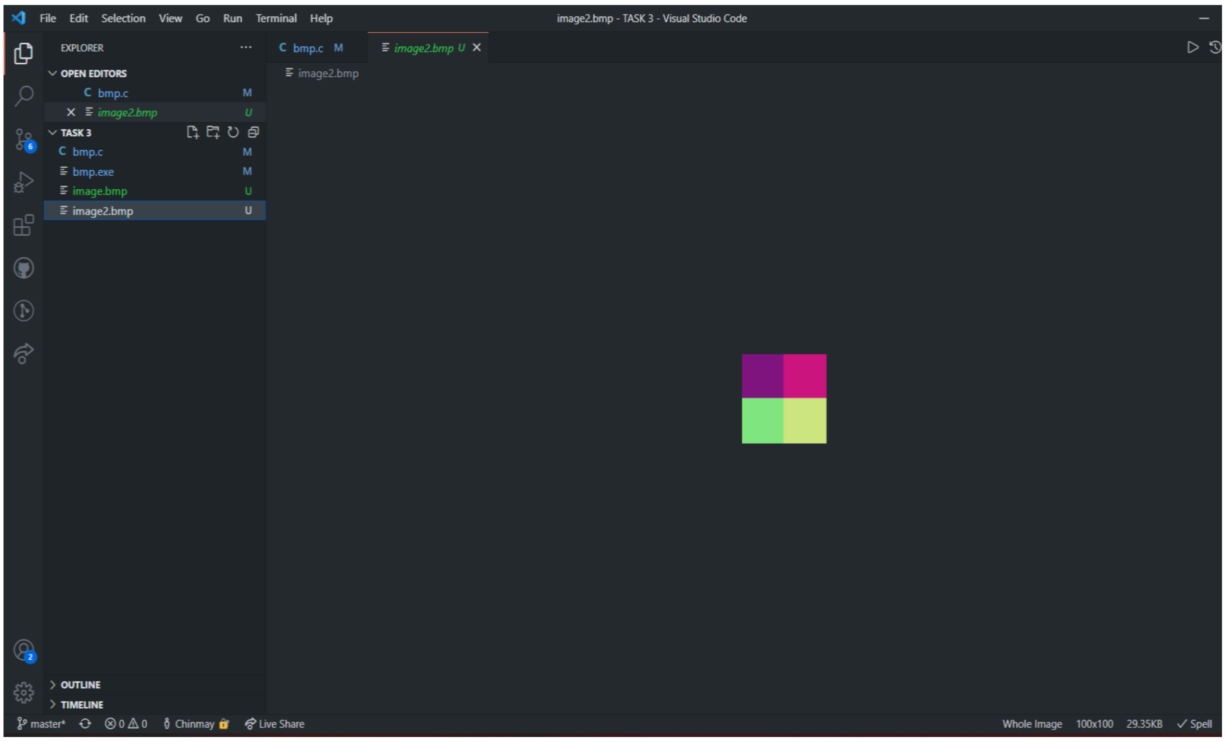
OUTLINE
TIMELINE
master* ⊗0 ⚠0 Chinmay Live Share
mahendra, 3 days ago via PR #1 Ln 133, Col 22 Spaces: 8 UTF-8 CRLF C ✓ Spell Win32

4. Input (Read) Bitmap Image



File Edit Selection View Go Run Terminal Help
image.bmp - TASK 3 - Visual Studio Code
EXPLORER
OPEN EDITORS
bmp.c M
image.bmp U
TASK 3
bmp.c M
bmp.exe M
image.bmp U
image2.bmp U
bmp.c M
image.bmp U
image.bmp
OUTLINE
TIMELINE
master* ⊗0 ⚠0 Chinmay Live Share
Whole Image 100x100 29.35KB ✓ Spell

5. Output (Write) Bitmap Image


File Edit Selection View Go Run Terminal Help
image2.bmp - TASK 3 - Visual Studio Code
EXPLORER
OPEN EDITORS
bmp.c M
image2.bmp U
TASK 3
bmp.c M
bmp.exe M
image.bmp U
image2.bmp U
OUTLINE
TIMELINE
bmp.c M
image2.bmp U
image2.bmp
master* ⊗0 ⚠0 Chinmay Live Share
Whole Image 100x100 29.35KB ✓ Spell