ECOAR lab – MIPS project winter 2018/2019

The black and white image is encoded using one bit per pixel. The oldest bit in the byte corresponds to the leftmost pixel in the image. The image line is aligned in the file to the nearest multiple of 4 bytes. In the black and white image we have following dependencies:
Number of bytes containing image pixels = (image width + 7)/8
Number of bytes in line (in file) = (((image width + 7)/8 + 3)/4) * 4

Information about image memory and important drawing parameters are stored in the imgInfo structure:

```
struct {
      int w, h;   // width and height of image
      unsigned char* pImg; // pointer to the image buffer
      int cX, cY; // coordinates of current drawing point
      int col;    // drawing colour (0 – black, 1 – white)
} imgInfo;
```

Implement the following functions:

```
imgInfo* SetColor(imgInfo* pImg, int col);
```

where

col new drawing colour (0 – black, 1 – white)

Function changes current drawing colour in imgInfo structure. Function returns pointer to the modified imgInfo structure.

```
imgInfo* MoveTo(imgInfo* pImg, int x, int y);
```

where

x new x (horizontal) coordinate of current drawing point

y new y (vertical) coordinate of current drawing point

Function moves current drawing point coordinates to (x, y). Function returns pointer to the modified imgInfo structure.

```
imgInfo* LineTo(imgInfo* pImg, int x, int y);
```

where

x new x (horizontal) coordinate of current drawing point

y new y (vertical) coordinate of current drawing point

Function draws straight line segment between (cX, cY) and (x, y) using Bresenham's algorithm. Function returns pointer to the modified imgInfo structure.
Of course, the only important challenge is the LineTo function. It is worth analyzing the Bresenham algorithm, which you can use to draw segments in the code graph_io.c or Wikipedia (https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm). The basic criterion that you must consider when designing and implementing a function (or functions) is the number of writes (and reads too) to memory. Of course, the point is to make it as small as possible.
What you can see in the example code (SetPixel calls - that is, drawing pixel by pixel, is not a satisfactory solution, at least not in all cases).
To build sample code use command:

```
gcc –m32 –fstruct-pack graph_io.c -lm
```