The PCX file format

Image files used by PC Paintbrush product family (those with a .PCX extension) begin with a 128 byte header. The remainder of the image file consists of encoded graphic data. The encoding method is a simple byte oriented run-length technique. When more than one color plane is stored in the file, each line of the image is stored by color plane (ordered red, green, blue, intensity), As shown below.

```
Scan line 0:        RRR...       (Plane 0)
                    GGG...       (Plane 1)
                    BBB...       (Plane 2)
                    III...       (Plane 3)
Scan line 1:        RRR...
                    GGG...
                    BBB...
                    III...       (etc.)
```

The decoding method is:

```
    FOR each byte, X, read from the file
        IF the top two bits of X are 1's then
            count = 6 lowest bits of X
            data = next byte following X
        ELSE
            count = 1
            data = X
```

ZSoft .PCX FILE HEADER FORMAT

Data type	Name	Description
UINT8	Manufacturer	Always 0x0A
UINT8	Version	PC Paintbrush version. In this exam always 5
UINT8	Encoding	1 = PCX run length encoding
UINT8	BitsPerPlane	Number of bits per pixel in each entry of the colour planes (1, 2, 4, 8)

UINT16LE	WindowXmin	Window (image dimensions):
		ImageWidth = Xmax - Xmin + 1
		ImageHeight = Ymax - Ymin + 1
		Normally Xmin and Ymin should be set to zero.
		Note that these field values are valid rows and columns, which is why you have to add one to get the actual dimension (so a 200 pixel high image would have Ymin=0 and Ymax=199, or Ymin=100 and Ymax=299, etc.)

Data type	Name	Description
UINT16LE	WindowYmin	
UINT16LE	WindowXmax	
UINT16LE	WindowYmax	
UINT16LE	VertDPI	Ignore in this exam
UINT16LE	HorzDPI	Ignore in this exam
UINT8	Palette[48]	Ignore in this exam
UINT8	Reserved	Ignore in this exam
UINT8	ColorPlanes	Number of colour planes. Multiply by BitsPerPlane to get the actual colour depth.
UINT16LE	BytesPerPlaneLine	Number of bytes to read for a single plane's scanline, i.e. at least ImageWidth ÷ 8 bits per byte × BitsPerPlane. Must be an even number. Do not calculate from Xmax-Xmin.
UINT16LE	PaletteInfo	Ignore in this exam
UINT16LE	HorScrSize	Ignore in this exam
UINT16LE	VerScrSize	Ignore in this exam
BYTE[54]	Padding	Ignore in this exam

Decoding .PCX Files

First, find the pixel dimensions of the image by calculating `[XSIZE = Xmax - Xmin + 1]` and `[YSIZE = Ymax - Ymin + 1]`. Then calculate how many bytes are required to hold one complete uncompressed scan line:

```
TotalBytes = NPlanes * BytesPerLine
```

Note that since there are always an even number of bytes per scan line, there will probably be unused data at the end of each scan line. `TotalBytes` shows how much storage must be available to decode each scan line, including any blank area on the right side of the image. You can now begin decoding the first scan line - read the first byte of data from the file. If the top two bits are set, the remaining six bits in the byte show how many times to duplicate the next byte in the file. If the top two bits are not set, the first byte is the data itself, with a count of one.

Continue decoding the rest of the line. Keep a running subtotal of how many bytes are moved and duplicated into the output buffer. When the subtotal equals `TotalBytes`, the scan line is complete. There should always be a decoding break at the end of each scan line. **But there will not be a decoding break at the end of each plane within each scan line (this means, for example, that a run could contain the last bytes of the R plane and the first bytes of the G plane, if they have the same values)**. When the scan line is completed, there may be extra blank data at the end of each plane within the scan line. Use the `XSIZE` and `YSIZE` values to find where the valid image data is. If the data is multi-plane, `BytesPerLine` shows where each plane ends within the scan line.

Continue decoding the remainder of the scan lines (do not just read to end-of-file). There may be additional data after the end of the image (palette)

1 bpp bilevel images

Bilevel images (black and white) are stored as 1 bpp, 1 plane images. Every byte contains 8 pixels from left (MSB) to right (LSB).

Truecolor images

24 bit images are stored as 8 bit, 3 plane images. 24 bit images do not contain a palette. Bit planes are ordered as lines of red, green, blue in that order.

256 indexed color images

Originally it wasn't possible to store palettes containing more than 16 colors in the PCX image file, so it is not available in the header. The 256 color palette is formatted and treated the same as the 16 color palette, except that it is substantially longer. The palette (number of colors x 3 bytes in length) is appended to the end of the PCX file, and is preceded by a byte with value 12 in decimal. To access a 256 color palette read to the end of the file and count back 769 bytes. The value you find should be a 12 decimal, showing the presence of a 256 color palette. A palette is a sequence of 256 triplets of bytes (R, G, B). The first one is the color to assign to pixels with value 0, the next one is for pixels with value 1, and so on up to the last entry which is for pixels with value 255.