CS 61C Project 2: RLE Representation Fall 1998

Administrative Details

This project is to be programmed in C. Your TAs will help you with details on C file input and output, memory allocation, and pointers, all of which are required to complete this program. It is due at Noon on Wednesday, October 14.

Storing Pictures

As you are well aware, pictures can be stored inside computers in various ways. Even a text file is a kind of picture. Other pictures are stored as arrays of colored dots or pixels.

Optical scanners are becoming quite inexpensive and allow you to read in pages of material, not as text, but as a map of individual bits, a scan-line at a time. Usually computers do not store this bit-map directly, but use some kind of encoding that is simultaneously more compact and faster to process. We introduce a simple alternative to a bit-map here. Imagine you have scanned a black-and-white (or 2-level, no color, no gray) image into the computer and have stored it (somewhat clumsily) as a file of characters. Below you see a small piece of such a scanned image, 67 by 115 pixels where we have printed the black pixels as "@" and the white pixels as blank. If you stand back, you might notice that it is the image of a digit 3. This was, in reality, scanned in at 600 dots per inch by a page scanner. (How large was it in real life?)

We want you to take a file with such information and produce a Run Length Encoded (RLE) version of the picture. In particular, the top line (line 114) will store the information (25,40), signifying that there is one segment of black dots including all the pixel positions from 25 through 40. For more variety, row 105 has two black segments, (10,24) and (38,47).

PART I

Your task is to take a "text picture" such as the one shown on page 3, or available in the file `lib/rle.txt`) and compute the length of the longest line (that is, the width), the number of lines in the file, and a list of all the lines. Each line in the file will be translated into a linked list of start and stop pairs. To be more specific, for the picture below you would produce a linked list of 114 line structures. Internally, each line should itself be a list of pairs `(start, stop, *pair)`.

Your program must be general enough to allow for any size picture, though as a practical matter you can assume all coordinates of pixels are between 0 and 32,000. There can be very few or a large number of RLE pairs on any line (16,000 if 32,000 bits are alternately black and white). You must allocate structures only as needed.

You should be careful that your program works for entirely blank lines, which are surprisingly common in normal text. A blank line has no pairs.

You must also write a `clean_up` program that returns all the allocated structures from a run of your program so that you can write a loop to read in any number of text pictures without permanently storing material in memory.

Your main program should call your program `rle_read` to read in a file, and then

(a) print on `stdout` the height and width of the picture.

(b) print the RLE encoding, line by line as:

```
line 0: ((start,stop),(start,stop), ...)
line 1: ((start,stop),(start,stop), ...)
```

...

(c) print a count of how many RLE pairs were needed for the picture.

(d) print a count of how many black pixels were encoded.

You should provide, for debugging purposes, a program to print the RLE encoding back as a text picture.

PART II

You should write in complete English sentences, a comparison of the following ways of storing black and white scanned pictures of text in a computer. Assume that whole pages of size 8.5 by 11 inches are scanned at 600 dots per inch, and you are scanning typical pages from a book. Come up with a figure on how many black-white and white-black transitions there are in a typical scan line. Using this information, consider these representations:

1. The text pictures as given, essentially a fixed array of bytes.

2. The text pictures, but using 0,1 bits, 8 bits per byte, instead of our clumsy @ and blank representation.

3. The RLE version of a picture stored in memory. (Be sure to specify how you are counting up the storage.)

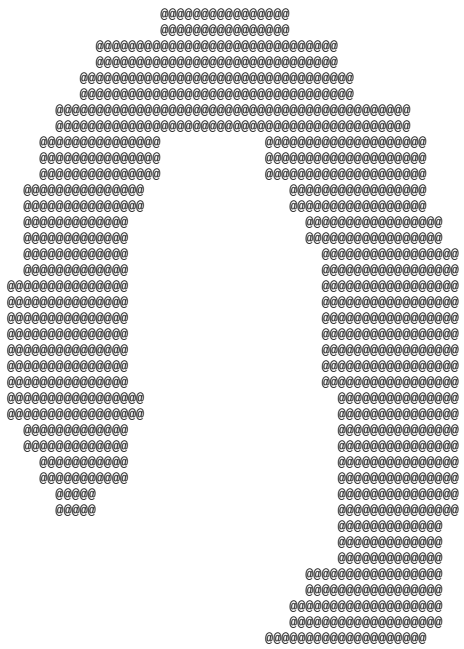
4. RLE pictures stored on disk in the following way: First a 16-bit unsigned integer N, the count of the number of RLE pairs in this "scan line". Then 2*N 16-bit unsigned integers representing the N start/stop pairs. Then the next line's count (or end-of file).

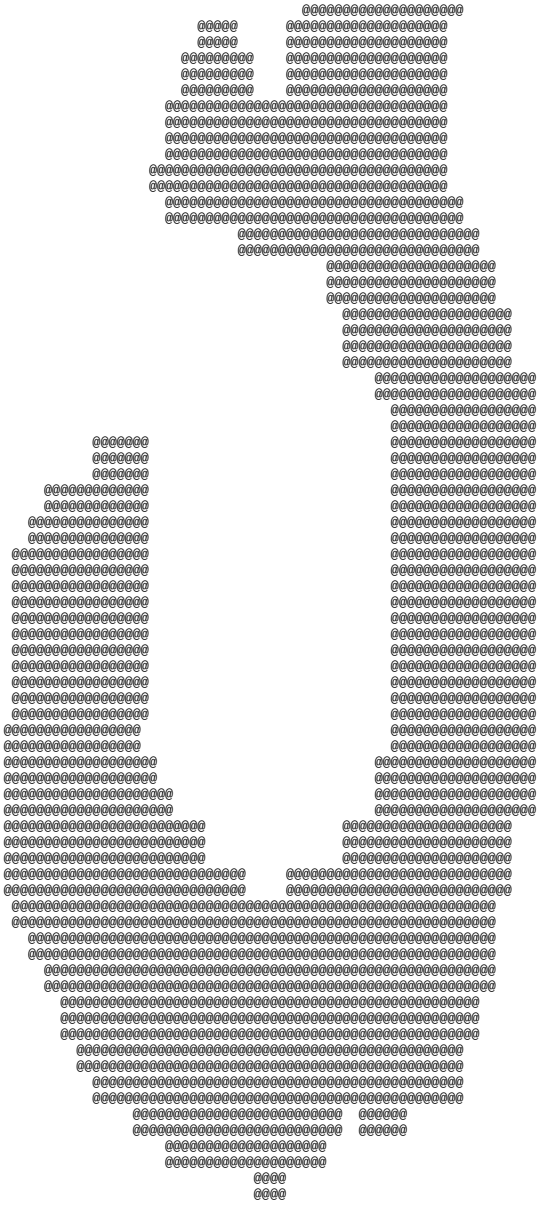
(II- A) Write up your comparison based on efficiency of storage in the computer memory or (for 1,2,4) on disk.

(II-B) Comment on speed. Speculate (or if you wish, you can even try some experiments to time this). For representations 1,2,3: How efficiently you can answer part (d): how many black pixels are there in the picture?

Here's some data

the top line is 114 , the last non-blank one is 0





Part III (optional)

You may, if you are curious, poke around on our HP machines, and learn about many advanced variations on static (non-animated) picture encoding. The xv program on UNIX systems provides GIF, PM, PBM, X11 bitmap, Postscript, JPEG; there is also a system called FlashPix. Then there are video encodings such as MPEG. Your favorite internet search engine may be useful in learning more.