Lab #1: Looking inside PostScript

This lab has a target due date of **Wednesday, January 23**. It counts three points.

1. What is PostScript?

PostScript is a "page description language"—a graphics language for printing text and pictures. It was developed by Adobe Systems, Inc. Its primary use is to be generated by programs to drive graphic output devices, but it is also human-readable. Some of its interesting characteristics:

- It is in ASCII text and is used that way; there is no binary version. A language that is meant to be used directly rather than being compiled first is often called a "script", as in the name "PostScript".

- PostScript uses "reverse Polish" or "postfix" notation: Rather than use `2+3` it uses `2 3 add`, with the operation written at the end. This notation explains the "Post" in "PostScript". (Of course, the name "PostScript" is a pun in that it's also an English word.)

- Because of the postfix notation, no parentheses are necessary in complicated expressions. For example, instead of `7*(2+3)` PostScript uses `7 2 3 add mult`. In evaluating this expression, the numbers are examined left to right, with results so far kept on a stack. Initially the stack is `7`, then `7 2`, then `7 2 3`. When `add` is reached, the top two numbers on the stack are replaced by their sum, making the stack be `7 5`, and when `mult` is reached, the remaining two numbers on the stack are replaced by their product, leaving `35`.

  It is also not necessary to use separators such as ; between commands; commands can be written on separate lines or strung together. Computer-produced files can be incomprhensibly messy, but files you make should be kept neat.

- Comments start with `%` and continue to the end of the line.

- The very first line starts with `%!`
  So if you send a file to a printer that knows PostScript (such as `popeye`), if the file starts with `%!` then it is interpreted as a PostScript file; otherwise it is just printed as straight text.

- As mentioned, PostScript is intended mainly for computers to talk to devices. Debugging information is minimal or absent! If you display an incorrect PostScript file, you get an error comment on-screen, but it may be obscure. If you send an incorrect PostScript file to the printer nothing happens (and maybe nothing happens for the *next* person to use the printer).

- PostScript is licensed by Adobe for use in particular printers, display software, etc. Therefore some printers are "PostScript printers" and others are not. PostScript printers have a built-in microprocessor, memory, and software for interpreting the PostScript language.

- To display a PostScript file named `myfile.ps` just double-click on it; the system should automatically open it using the Ghostview program (= GSView).

  To print the display on paper in the lab, just print it like any other file. However, it is dangerous: If there is a bug in your file it may make the printer hang for everyone! For safety, display the file on your PC first.

- On the other hand, to see a file as *script* (showing the PostScript commands themselves), on the NT system use the `Notepad` editor, found under Accessories on the START menu. To *print* a file as script, you need to prevent the printer from seeing the first line with `#!` , so one way is to make a copy, edit the copy to delete the first line, and then print the copy.

- When you draw lines you first say where the lines go and then use the command `stroke` . To draw a closed figure, you can draw point to point until the last point and then use the command `closepath` along with `stroke` . A closed path is essential to draw a shaded figure.

- The basic unit for drawing is 1/72 inch. However, other units can be used, either by scaling every coordinate each time it is mentioned, or by changing to a coordinate system with a different scale.

- Initially the origin for drawing is in the lower left corner of the page. However, when using ghostview (GSView) you may find that the lower left corner is below the current window, so scrolling may be necessary to see things drawn near the origin.

- Sometimes a software package will output a PostScript file without the final `showpage` . That should not be the case in this assignment, but you should be aware of the possibility. Ghostview seems not to care about this but a PostScript printer will sit there and do nothing.

- There are some manuals on PostScript, but they are not needed for this assignment. If you are interested, I can show you my copies, and there are possibly copies in the UCLA Store General Books department.

- You will often see references to "Encapsulated PostScript". This is the same as a PostScript file except that near the beginning of the file there is extra information about the ranges of x and y coordinates used (the "bounding box"). This information enables programs such as TeX to incorporate the file with other output, e.g., a diagram in a page of text.

Philosophy of this assignment: In the future, you will often run into a situation where there is something complicated that you need to go in and alter without having full knowledge—perhaps it will be a big program written by the employee you replaced, or a smaller program written in a special language with an obscure manual. In confronting such a situation, try to look for pieces of examples that you can follow.

This assignment is somewhat similar, since you are to write and modify PostScript using examples instead of the manuals. However, a few comments are provided in the examples.

Task A: "Pure" PostScript. Some example files named `demo1.ps`, `demo2.ps`, etc., can be found in the handout directories on both the NT and UNIX systems, whose locations were given in the previous paper handout.

The task is to use the ideas in these demos to make some Postscript file that is distinctly your own. If you can, do something interesting. If you're too pressed for time, do something more rudimentary. One or more interesting submissions may be posted for everyone to see.

Because debugging is so difficult, proceed in steps, starting with something that is almost like a demo and modifying it, while checking it with `ghostview`.

Submit your file as described in the previous handout. Name it `lab1A.ps` .

Task B. Landscape versus Portrait modes

PostScript initially assumes that the coordinate system has its origin at the lower left corner of the page when the paper is held with the longer way vertical (portrait mode). Suppose that the picture you want to draw is wider than it is long and you want to print it that way (landscape mode). The way to do this is to rotate and translate so that your data is changed to portrait mode (although it will be on its side when displayed on the screen), print it, and then hold the printed copy the landscape way.

In the handout directory you will find a file `chi.ps`, done by a former student in this course. It is presently in portrait mode. Put in extra changes of coordinates to make the picture come out turned sideways on the screen, so that if printed on paper it will be in landscape mode. A good place to to do this is marked in the file. You should decide what combination of translations and rotations is needed. Don't forget to change distances to inches if that's what you need. Notice that you can't just rotate the picture from a corner, since you'll be rotating it right off the screen; a translation will also be involved.

Submit the modified file, named `lab1B.ps` .