CS400: Problem Seminar
Fall 2000
Assignment 1: Hello World!

Handed out:	Sept. 6, 2000
Due:	Sept. 13, 2000 in class

It is important to learn various tools of the trade lest programming, writing, collaborating, and life in general become very inefficient. In my experience, it is almost always a good idea to master these utilities—the more you know the easier life gets. You should gain familiarity with basic tools for editing, programming, debugging, managing large programs, finding information, producing papers, managing references, *etc*.

Your first assignment is therefore to familiarize yourself with the Department's computer systems and tools. These include the machines themselves, the Unix environment, text editors, document formatters, and programming tools. Specifically, you must do the following:

- Get to know Unix.
- Create a web page for yourself.
- Produce a "hello world" program in six different programming languages.
- Write a short report describing your efforts and results.

The remainder of this document describes these tasks in more detail.

The long "Setting Up Your Computing Environment" page at `http://www.cs.rochester.edu/u/jason/400/setup.html` has a lot of useful advice. Your first step should be to look through that document carefully. Among other things, it has enough resources to get you started on this assignment.

If you find additional helpful sources, please share them with your classmates. (You can send them to the class mailing list, `cs400`.) If you already know everything there is to know about Unix (or whatever), please help your classmates. This is a learning experience, not a competition.

One warning: Do not leave the report to the last minute! You should write up what you did regardless of whether you were able to do it all or not. A good strategy is to create a skeleton document early on, and fill it in as you do the work.

1 Get to know Unix

Everything you do for the next four or five years will be done using Unix, so it is worth your while to make the relationship as productive as possible.

The "Setting Up" page suggests some resources to get you started with Unix. Try an online Unix tutorial! In order to do the rest of this week's assignment, you will at least need to be comfortable manipulating files and directories. But you may also want to start configuring your environment to your liking, learning some communications tools, and organizing your online life. The "Setting Up" page has suggestions about all this.

2 Create a web page

For this assignment, you need to create a "home page" at URCS and add some content to it. You can describe yourself, your interests, and so on. If you want to include a picture of yourself, Marty Guenther has a digital camera that can take your picture. Your home page should be proper HTML, and should have the appropriate URL as described on the "Setting Up" page.

3 Hello World programs

This part of the assignment will get you used to some of the various programming languages and tools available on our systems. For each language listed below, you need to create a program that:

1. Prints "hello world" to standard output;
2. Reads a line of text from standard input;
3. Prints that text to standard output.

While you may want to experiment further for your own edification, these programs should take no more than a few lines of code in any of the languages. Write good code—e.g., try not to crash on ill-formed or overly long lines—but don't go overboard on functionality. You'll get your chance at that soon enough...

3.1 Programming for Utilities: Perl and Shell

The Unix environment is designed around the idea of tools: tools are provided and you are expected to build on these and create your own tools to solve new problems. The two most common building blocks are the shell and the Perl programming language. Your "hello world" program should be very, very simple in either of these.

For your shell program, you may use the shell of your choice. You should also do a Perl program.

3.2 Imperative programming: C or C++

You can use either C or C++ for this part of the assignment. If you're only familiar with one, you might try using the other. You should create a `Makefile` so that you can use the `make` command to compile the program.

3.3 Functional programming: Lisp

Unfortunately, the Hello World task is rather imperative, so it rather misses the point of a functional language like Lisp. You should use the I/O facilities of Common Lisp for your program.

3.4 Object-oriented programming: Java

It's fine to write an ordinary Java application that reads standard input and writes standard output. If you prefer to write an applet with a graphical user interface, be my guest.

3.5 Logic programming: Prolog

To exercise Prolog's logic programming abilities very slightly, we'll change the task a bit for this language.

Assume you have the following knowledge (that is, put this in your program):

```
world(earth).
country(usa).
country(canada).
world_moon(luna).
crater(tycho).
world(mars).
chasm(valles_marinaris).
```

Your program should print "hello world", input a line (or just a term followed by a period), then print whether the input was a world or not, as defined by the above set of facts.

4 Write a report

The most important thing you will do in graduate school is present your ideas in writing. You will need to write an area paper, a thesis proposal, and, of course, your dissertation itself. You will likely be writing tech reports, conference papers, journal articles, and so on through your graduate career. So it makes sense to start learning how to prepare written documents as soon as possible.

For this part of the assignment, you must write a short report describing your efforts on the other parts of the assignment. While there may not be much to say, your report should be well-written, proofread and spell-checked, and formatted using LaTeX. If you don't know LaTeX, you must learn it (or enough of it to write this simple document).

You should describe what you did for each part of the assignment. Also indicate where the programs live, include their source code, and provide traces of their execution. Finally, just to prove you can really use LaTeX, you should include a formula and a drawing. The formula and drawing don't have to mean anything, but they should be at least as complicated as, say, the quadratic formula and a line drawing of an ice cream cone.

I recommend that you figure out how to use the `auctex` package for Emacs. This package makes common commands readily available, and allows you to run LaTeX and related programs from within Emacs.

To check spelling in your document, use `ispell` (within Emacs, type "`M-x ispell-buffer`").

One more thing: We waste a lot of paper in the department. Learn to use the `duplex` command to print files double-sided. You should hand in your report printed in this manner (**don't** simply photocopy it from single-sided—learn to use the tools).

Good luck and have fun!