Term Project and Homework 1

Homework 1 is due Thursday, Feb. 1 (by 5:00 PM). The term project is due Monday, May 7 (by 5:00 PM).

The minimum percentages necessary for each letter grade on a homework are as follows: A = 85%, B = 75%, (straight) C = 67%, C- = 65%, D = 55%. Note that exactly 85% would be an A- grade for a homework. The same standard will apply to the term project.

1 Term project

(100 points) Your term project is to write a short paper that critically evaluates and suggests possible improvements to a sufficiently language-like computer system (see below). The first problem of the homework below requires you to negotiate with the instructor the particular system you will evaluate. This negotiation should be completed by Feb. 1, and will ensure that you are choosing a suitable system to evaluate, and that no one else in the class is choosing the same system.

Your evaluation should be no more than 10 (typed) pages long. (You may use extra pages for figures, examples, or diagrams if you wish.) Please describe the system in general terms, identifying the interesting features, then critically evaluate the design as a whole (without getting bogged down in details), making concrete suggestions for improving the system where appropriate. You may find it helpful to discuss the design in terms of language design principles, but do not forget other issues such as efficiency, and only focus on what seem to be the most important principles for the system (i.e., what is really good and really bad). You may also find it helpful to make analogies with the languages we study in class. Please give a bibliography of books or articles that you have consulted, including those that describe the system. You are responsible for finding documentation on the system you choose.

Some of the points for this problem will go to how well your description is presented. You should check over your answer carefully before handing it in. Do not assume that the reader understands the system you are discussing.

A computer system will be considered to be "sufficiently language-like" if it meets all of the following criteria:

- It has some form of names that can be bound to values (i.e., it has variables).
- It has subparts that are separately documented (i.e., it has several commands or tasks that it can perform).
- Its user interface *cannot* be described with a regular grammar (note that the user-interface involves both commands and any data file formats).
- The system is not primarily thought of as a programming language.
- It has been implemented (you do not need to have access to an implementation).

Most systems that are described with a book should qualify under these guidelines. However, make sure you get a book that describes some specific system, instead of a book about a topic (such as database systems).

Some suggestions for systems follow (but don't feel limited by these). Check in the library or the bookstore to find reference materials.

- Algebraic manipulation systems such as Mathematica, Macsyma, Maple, etc.
- Database management systems, such as Ingres, DBII, etc.
- Editors, such as Word Perfect, Word Star, Emacs, etc.
- Spreadsheets, such as Lotus 1-2-3, etc.
- Statistical software packages such as SPSS, etc.
- Document preparation systems, such as $\TeX$, Scribe, InterPress, etc.
- Operating system command languages, such as IBM 360 JCL, etc.
- Specification languages, such as Z, Clear, etc.

2 Homework 1

1. (5 points) Negotiate, by Unix mail, your term project. That is, send a mail message to `leavens` on Zippy describing your term project, wait for a reply, and follow the actions of the reply (which may require you to choose a different project, resubmit your message, or accept your proposal). Be sure that your mail messages have a subject line (such as "proposed term project"), include your full name, and are checked for spelling and grammatical errors (compose the message using an editor). Messages not meeting these guidelines will have to be submitted again.

2. (7 points) Write a complete pseudo-code program to read in data cards (until a +0 000 000 000 flag card), add up the numbers on the cards, and print out their standard deviation. The standard deviation of n numbers $x_1, \ldots, x_n$ is defined by the following formula, where $\overline{x}$ is the mean (or average) of the x_i.
$$\sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}}.$$

3. (5 points) In the design of MacLennan's pseudo-code (see chapter 1), there is no "inverse operation" that corresponds to the move operation in the way that subtraction corresponds to addition or that division corresponds to multiplication. Is this design (i.e., having no meaning for the operation code -0) a reasonable one? Why or why not?

4. (5 points) Choose a single pseudo-code instruction from MacLennan's langauge that you believe would cause the least problems for the pseudo-code language if it were omitted. Justify your answer and also state whether it would be a good idea to omit your choice from the language.