Homework 2: Project Start

See the syllabus and listen in class for the due dates.
In this homework you will get a start on your semester project.
If you wish, you can work in groups, and I recommend that for the semester project. However, be sure to follow the process described in the course's grading policy if you work in groups.
Turn in this homework on paper during class.

Read chapter 1, and if you have time, chapter 2, of our textbook: *Principles of Program Analysis* [1].

1. (30 points) [Concepts] For this problem, the statement of a *program analysis question* is a careful specification of what an analysis will determine at each program point. For example, section 2.1.1 of our textbook [1] says that "the available expressions analysis will determine:"

   > "For each program point, which expressions *must* have already been computed, and not later modified, on all paths to the program point."

   Other examples appear at the beginning of sections 2.1.2, 2.1.3, and 2.1.4. Note that these statements often rely on auxiliary definitions.

   For your own semester project, list the most important program analysis question(s) that the project will have to answer. You can list up to 3 of these for your project. If you have others, you might want to write them down for yourself, but only hand in the most important three questions for your project.

2. (100 points) Write and test a parser and the construction of abstract syntax trees for the programming language (fragment) that you will be using in your semester project.

   We recommend that you use JastAdd (see `http://jastadd.org`). For working with JastAdd, you should read the on-line documentation, especially the reference manual. Also, it's useful to start with an existing JastAdd sample project (see `http://jastadd.org/projects`), such as PicoJava, and modify it to suit your language.

   You can use our WHILE language as a sample project also. The Unix command for anonymous checkout is:

   ```
   svn checkout http://refine.eecs.ucf.edu/svn/proganalysis/WHILE/trunk WHILE
   ```

   and the URL `http://refine.eecs.ucf.edu/svn/proganalysis` can be used with tools like Eclipse, but check out the WHILE/trunk if you are working with Eclipse. If you have permission problems with SVN, browse to `http://refine.eecs.ucf.edu/gf` and get a login, then send the instructor an email about the problem.

   We recommend that you use a scanner generator (such as "flex") and a parser generator (such as "beaver"), since this will allow you to change your language more easily as your project progresses. To avoid parsing troubles, we recommend that you use reserved words to uniquely identify each statement (or expression, etc.). Note that the syntax of C and C++ have notorious parsing difficulties (to some extent inherited by Java), but many of these can be avoided by adding some additional keywords. If you have trouble with the grammar of your language, or with ambiguity in your grammar, see the instructor or send an email.

   For this project, write both a parser and an aspect that does unparsing. Also you will need tests to demonstrate that your parser and unparser work properly. Hand in your source files (but no generated files!) on paper.

References

[1] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Analysis*. Springer-Verlag, 1999.