# HOMEWORK 1

CS 322 Compiler Construction
Winter Quarter 2005

**Due : Tuesday, March 1 at 2:00pm**

1. Consider the following strategies for maintaining symbol tables:

   A. Maintain one symbol table per scope, implemented as a linked list of pointers to symbol table entries. When looking for a variable, we first traverse the list corresponding to the current scope, then the immediately outer scope etc.

   B. Maintain a single symbol table, implemented as a hash table. Each identifier is associated with a linked list of entries, ordered by scope (most recent first).

   C. Maintain one hash table per scope (similar to A except we use hash tables instead of lists)

   Now, suppose that each scope has 10 entries and that each scope incurs 100 lookups, 90 to items of its own scope and 10 to the next outer scope. Assuming perfect hashing compare the three strategies in terms of the number of entries probed in a scope. If we now consider table/list setup and maintenance cost will your answer change and how?

2. An alternate heap allocation strategy is to always allocate space from the largest available free block. How does this compare to the best-fit and first-fit strategies?

3. In dynamic scoping, the current "binding" between a variable name and its value is the one that was most recently encountered during execution. The class notes give an example of how we can use this to customize a procedure as well as how we can achieve the same effect without using dynamic scoping. Nevertheless, dynamic scoping *is* used in languages such as Perl and LaTeX as well as with Unix environment variables. Could we do without dynamic scoping for environment variables and instead use parameters or static variables? If not, why not? If yes, how, and would it be better or worse than using dynamic scoping?

4. Some languages allow the programmer to use functions in the initialization of local variables, but not in the initialization of globals. What reason could there be for it?

5. The cost of a variable referencing mechanism is defined as the number of loads and stores imposed by the mechanism to maintain its data structure and access variables. Discuss the conditions under which a Display mechanism will be cheaper than a static link mechanism and vice versa.

6. Some languages require the compiler to use short-circuit evaluation. The programmer can use this to his advantage, to check for safety. For example, the code `a!= 0 && b/a>0.5` will avoid a potential division-by-zero error. If a language definition does not specify short-circuit evaluation for boolean expressions, should the compiler do it anyway as a form of optimization? Justify your answer.

## Late submission

The 72-hour policy does not apply to written homework. If you submit late, you will be penalized by 25% per day.