STUDY SHEET FOR FINAL EXAM

There will be "what-if" questions where you reason from knowledge about your compiler in a "what-if" scenario. Example: what if we wanted to include a const keyword or a for keyword? You will need to know how you have to change each phase of the compiler to get these to work or to disable something. It mostly covers things we covered in class that were not on the first test. Below are some topics and example questions to look at.

Good Luck!

---
what are attribute grammars?
what are attributes?
what are examples of attributes?
understand the table on p263
understand the table on p270
understand the grammar on p265
understand the table on p266
understand the figures on p273 (DEPENDENCY GRAPHS)

what are synthesized attributes? (see p277)
what are inherited attributes?
what is an attribute equation?
why is order of traversal of the nodes of a tree important?
understand algorithms for passing information up, down and across a tree.
(see section 6.2.2)
How does having multiple passes help in attribute evaluation?
Understand example 6.18

Symbol tables
What kind of information is stored in a symbol table.
Be specific, not general.
What is bad about algorithm in fig 6.13?
what is meant by scoping rules?

The following two declarations are not allowed in C-
but are allowed in the language D-.
```
int kate;
int kate() {
}
```
What specific changes would we make to the use of the symbol tables in D- to make this allowable in D-?

How would you include a for statement into C- if you are allowed to produce any tree nodes you want in the parser phase?
How would you include a for statement into C- if you must use only the for node rather than rewriting the tree?

Why does `x = y--3;` give an error rather than go to the code for `x = y - -3`?
Explain in terms of both the scanner and parser.

They symbol table is used to help generate a variety of semantic errors that are discovered by looking up in the symbol table. Give two different cases.

What is mutual recursion and why is it a problem in
C- and specifically address the role of the symbol table in this
problem. Suggest one way in which this could be fixed in C-

what is a variables lifetime?
what is lexical scoping?
what is a simple type?
In more complex languages like C++ or Ada the user
can create their own types. This creates the need
for structural equivalence. When are two things
structurally equivalent?
how is type checking handled for + and for = in C-?
Discuss the pros and cons of allowing coercion to occur between
each type and any other type in a language

How does scoping for local procedures work?

what is backpatching?
how does backpatching work with the if-then-else statement in c-
be complete and concise
how is break backpatched in c-?
how is data memory organized for c-? be sure to
show where reg 0 and 1 point and where the frames, globals,
old frame pointers, return addresses are. make a clear diagram.

how do static variables work in c-?
how are arrays referenced in c-?

What is a basic block?
what is constant folding
give two reasons why inlining is an important optimization technique
what is a loop invariant?

Where is the hidden loop invariant here:

```
     for (i=0; i<10000; i++) {
          for (j=0; j<10000; j++) {
             x[i,j] = y[i,j] + z[i,j];
          }
     }
```

What is the main advantage of loop unrolling.
Why is it easy to optimize tail recursion? In what way is it optimized?
be specific and complete