**Introduction**

**Master**: the phases of a compiler.

**Understand**: what is a compiler. All the concepts about the compiler stages

You Should be familiar with each stage of the compiler, and describe the whole process in graph or text.

**Lexical**

**Master**: Write regular expression, the transition from regular expression to NFA, then to DFA, DFA minimization and the construction of scanner

**Understand**: Concept of regular expression, NFA, DFA

Be familiar with the expression of R.E.. Know the difference of R.E. and C.F.G, can translate each from the other. While given a R.E, know how to draw the NFA and DFA, and how to minimize it. Master the method of transition table and subset construction.

**C.F.G.**

**Understand**: Context-free grammar, Derivation, Parse tree, Abstract syntax tree, ambiguous grammar. Total under all the concepts.

Be familiar with the grammar of C.F.G, know how to make a left/right most derivation. Given a set of grammar and an expression, can draw the parse tree and abstract syntax tree*.*

Know how to determine whether a grammar is ambiguous by examining whether its syntax tree exhibits ambiguity.

Be familiar with grammar rewriting, including ambiguity elimination through priority, associativity, left - recursion, left - factoring, etc.

**Top-Down Parsing**

**Master**: LL(1) grammar, Construction of Recursive-descent parsing, LL(1) parsing, Computing First set and Follow set

Given the C.F.G, know the left factor and left recursion removal, and know how to compute its first/ follow set, know to judge if it is a LL(1) grammar，and know to construct the parsing table and perform derivations.

**Bottom-Up Paring**:

**Master**: LR(0) parsing, SLR(1) parsing

**Understand**: Right sentential form, Viable prefix, Handle

It must be easy of you to write out the right sentential form of a given C.F.G and expression, and figure out the variable prefix and handles. Know the difference of LR(0) and SLR(1).

Given DFA，you can figure out the difference between LR(0) and SLR(1), and complete the LR(0) or SLR(1) parsing table according to whether the conflict exists.

**Semantic**:

**Master**: Dependency graphs, Algorithms for attribute computation

**Understand**: Attribute grammar, Synthesized and Inherited attributes. S-attributes, L-attributes, chained symbol table, scope checking, type checking

Know the task of semantic analysis, and the implementation method of it. Know how to compute the attribute values by given equations. Know how to draw a dependency graph, and how to label out the attributes or values to a parsing graph.

Master the symbol table construction based on pointers, and perform semantic checking of variables on such symbol table

**Intermediate Code Generation**:

**Master**: Intermediate code generation for basic structures：Three-address code;

TAC generation about control structure, expression, declaration and array reference

Given context-free grammar and Attribute grammar, can translate the source code into TAC by SDT, which means you can perform the label-attribute calculation by draw a syntax tree and dependance graph, and generate TAC by traversing this syntax tree based on attribute equations

考试题型 1）选择题 10题共20分；2）解答题 80分

题目顺序与难度无关。先做会做的，切勿在一个地方卡死，不要简单问题复杂化。