CS 516 Compiler Design Course Outcomes

Each course outcome is followed in parentheses by the Program Outcome to which it relates.

Parse tree construction - Construct a parse tree, or explain why no parse tree exists, given a BNF grammar and a string over the appropriate alphabet. ()

Lexical analyzer implementation - Implement a lexical analyzer from a specification of a language's lexical rules. ()

Eliminate square and curly braces - Translate a BNF grammar that uses "[ ]" notation and "{ }" notation into an equivalent grammar with no such notation. ()

Compute FIRST set - Compute the FIRST set for a BNF grammar. ()

Compute follow set - Compute the FOLLOW set for a BNF grammar. ()

determine FIRST intersect FIRST constraint satisfaction - determine if a BNF grammar satisfies the constraint on intersection of FIRST sets required for single-symbol-lookahead, top-down, lookahead parsing ()

determine FIRST intersect FOLLOW constraint satisfaction - determine if a BNF grammar satisfies the constraint on intersection of FIRST and FOLLOW sets required for single-symbol-lookahead, top-down, lookahead parsing ()

check for left recursion - determine if a BNF grammar satisfies the constraint on left recursion for single-symbol-lookahead, top-down, lookahead parsing ()

fix simple constraint violations - fix simple violations of constraints that preclude single-symbol-lookahead, top-down, lookahead parsing ()

construct parser - design and implement a single-symbol-lookahead, top-down, lookahead parser from a BNF grammar ()

design symbol table - design a symbol table format for the language defined by a BNF grammar ()

implement symbol table construction - enhance the parser to construct a symbol table as it parses an input ()

formulate semantic tests] - determine the specific semantic tests, e.g., type analysis, to perform on an input to the parser for a BNF grammar and enhance the parser to perform that analysis ()

implement semantic testing - enhance the parser to perform semantic tests as it parses an input ()

design code generator - design a syntax-oriented translation, into a specified intermediate code language, for the high-level language defined by a BNF grammar, and enhance the grammar.s parser to perform that translation. ()

implement code generator - enhance the parser to perform translation into the intermediate code language as it parses an input. N.B. the code generator need not generate target code from source code containing procedure/function calls ()

construct dynamic run-time stack - draw the dynamic structure of the run-time stack when target code containing procedure/function calls is executed. ()

Apply code optimizations - apply simple intermediate code optimizations ()