Date :
Page No.

CP

Pract. 1

Q. Role of lex in Compiler Design

→ lex is a tool used to generate lexical analyzer
↓
converts sequence of characters into tokens for ~~pass~~ parsing.

Parsing → Compiler checks syntax of code, ensuring it sticks to language grammar rules.

Q. What is YACC

→ Yet Another Compiler Compiler
→ tool used to generate parsers based on Content-free Grammer

Parser → is a compiler used to break data into smaller elements coming from lexical analysis phase.

→ Takes input in form of sequence of token & produce output in form of parse tree.
→ Two types :-
Top-down
Bottom-Up

Content Free Grammer → It's a set of rules used to describe structure of a language
Consists of 4 ~~things~~ tuples
Variable (V) Production Rule (P)
Terminals (T) Start Symbol (S)

Date :
Page No.

Q, What is token ?

→ It's a basic unit or building block of source code.
→ It a category of meaningful symbols that compiler recognizes & processes during compilation process.

Eg → int x = 5 ;

Types
- Keyword — int
- Identifier — x
- Literals — 5
- Operator — =
- Punctuation — ;

Lex handle lexical Analysis
Yacc " Syntax Analysis

Q, Output of lex & YACC → two files — y.tab.c
                          — y.tab.h (headerfile)

→ Generates a C program (lex.yy.c) that implements the lexical analyzer

yy is naming convention used by lex for variables & functions that lexer uses.

Q, How lex interact with YACC

→ lex generates tokens, which are passed to parser created by yacc to analyze syntax

Date :
Page No.

Pract. 2

Q, Grammar in content of CD

→ It is a set of production rules defining the syntax of a language

Q, Significance of start symbol in Grammar.

→ It is simply the first symbol in grammar to generate strings in a language.

Q, What are terminals & Non terminals

→ Terminal are the basic symbol that appears in input & cannot be further broken down (like keywords)

→ Non terminal are symbols used in grammar to represent structure that can be expanded into terminal.

→ Non terminal are used to build language from terminals.

Q, How to check if string belongs to grammar ?

→ By constructing a parse tree

Q Diff b/w LL & LR grammar

→ LL grammar are parsed from left to right using leftmost derivation

→ LR grammar are parsed using rightmost derivation in reverse.

Date :
Page No.

Pract. 3

Q, Diff blw keywork & identifier

→ Keyword have predefined meaning → Eg = Int
Identifier are names defined by user → Eg = CalculateArea

Q, Role of strstr() in c

→ function finds the first occurance of substring in string

Pract. 4

Q What is left Recursion in grammer.

→ Left Recursion occurs when a non-terminal refers to itself as leftmost symbol in one of its production. Eg ⇒ $A \rightarrow A\alpha | \beta$

Q, Why left recursion problematic in parsing?

→ Bcoz it causes infinite recursion in top-down parsers like LL parsers.

Q, Diff blw direct & indirect left recursion
DLR ILR

⇒ DLR occurs within a single production rule

→ ILR involves multiple rules.

Q, How Left Recursion removed ⇒ By Rewriting grammer using Auxiliary non-terminal

* LR parser can handle left recursion

Date :
Page No.

Pract. 5

Q, What is left factoring ?

→ It's a grammar transformation technique to eliminate ambiguity by factoring out common prefixes in production.

Ambiguity → a situation where a particular piece of code can be interpreted in multiple ways

Q, Why left factoring necessary ?

→ It helps eliminate ambiguity & improve efficiency of parsing algorithm.

Q, If left factoring not performed ?

→ lead to parsing errors or ambiguity in grammar.

Pract. 6

Q, Basic operation of Stack

→ Push, Pop & Peek

Q, Some application of stack in compiler

→ Parsing, function call management.

Q, what is stack overflow ?

→ Occurs when more elements are pushed than stack can hold.

Date :

Page No.

Q, How stack diff. from queue ?

→ Stack follows LIFO Last In first out
Queue follows FIFO First In first out