PART A

Discuss significance of lexical analysis.

What is LEX compiler.

Write regular expressions for the following tokens: a) Integer literals b) Identifiers

Discuss about brute force parser

List the advantages and limitations of bottom-up evaluation in the context of syntax-directed translation

Differentiate between synthesized and inherited atibutes.

List the types of errors encountered during the compilation process, focusing on lexical and syntactic phase errors.

What are the syntactic phase errors

Briefly discuss the storage allocation strategies

What is loop optimization. Explain with example.

Give the answer of following questions:

Name different phases of compiler.

Define the term Finite Automata.

What is recursive descent parsing ?

Differentiate between top down and bo up parsers.

List error recovery techniques for lexica analysis.

What semantic analysis does ?

What is Indirect Tuples ? I

What is Symbol Table ?

What is Context Free Grammar ?

What is Syntax Tree ?

What is Assembler ?

List tools used for compiler construction.

What is Yaco?

What is Lexical Analysis ?

List top down parser.

What is context free grammar?

What is synthesized attributes ?

What is symbol table ?

List various errors occur in compilation process.

PART B

Discuss the process of lexical analysis in compiler construction. Explain how tokens are specified and recognized

What are the main phases of a compiler? Briefly describe each phase

Compere and contrast top-down parsing and bottom-up parsing techniques.

Explain the concept of 1L(1) parsing. How is it different from LR parsing?

Discuss syntax-directed translation schemes and their implementation.

Discuss common sources of semantic ambiguities in L-Attributed Definitions

Explain the three address code implementation techniques with suitable example

Explain the process of run-time storage management in block-structured languages.

Describe the concept of peephole optimization.

. Dives the challenge and considerations in implementing simple code generator

Discuss the importance of input buffering in the design of lexical analysers.

Consider the following grammar:

EE+TTT->T*FFF> (E) | id

Create a parsing table for the above grammar, choose any parsing technique (eg.. 1.1(1). 1.R().ete

Write Syntax directed translator to implement calculator as well as draw the syntax tree for it

Describe symbol tables, their operations, and implementation

Describe the process of intermediate code generation. Why is it important in compiler construction

Briefly explain the phases that constitute front end of a Compiler.

What is the use of first and follow functions. Calculate the first and follow functions for the given grammar:

S -> A

A -> aB / A * d

B -> b

C -> g

What are the rules for eliminating left recursion? Consider the following grammar and eliminate left recursion

EB+ T/T

TT F/F Fid

Differentiate between DFA and NFA. Draw a DFA for the language accepting strings starting with "ab' over input alphabets ∑ = (a, b).

How shift reduce parser works? Considering the string '10201', design a shift-reduce parser for the following grammar:

SOSO1S1 | 2

What is three address code mechanism? Explain in brief by taking suitable example.

Consider the following grammar

EEAE id

A → + x

Construct the operator precedence parser.

Explain the working of YACC Compiler

What is S-attributed and L-attributed SDT's in syntax directed translation ?

Explain any two error recovery techniques in brief

Discuss in detail the various phases of a compiler with suitable example.

What is ambiguous grammar? How can it be converted into unambiguous grammar ?

Let G be a Context Free Grammar for which the production rules are given below:

S -> aBbA

A -> alaS/bAA

B -> blbS|aBB

Drive the string 'aaabbabbba' using the above grammar (using Left Most Derivation and Rigla most Derivation)

opilation

What are the problems associated with Top Down Parsing? Also write the production rules to eliminate the left recursion and left factoring problems.

Consider the following grammar:

E-> E+TIT T-> TFF F-> F alb Construct the SLR parsing table and also parse the input "ab+a".

piler

be

Write short notes on the following:

(a) Synthesized attributes

(b) Inherited attributes

Define an augmented grammar. Construct the LR(0) items for the following Grammar: SLR.

What is three address code and its importance ?

Convert the given expression:

a(b+c) into three address code

Explain the working of LL(1) parser.

(b) What is operator precedence grammar