18CS61

USN [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

Sixth Semester B.E. Degree Examination, Dec.2023/Jan.2024
System Software and Compilers

Time: 3 hrs.   Max. Marks: 100

Note: Answer any FIVE full questions, choosing ONE full question from each module.

Module-1

1 a. Explain registers, instruction formats and addressing modes of SIC/XE architecture.
   (08 Marks)
   b. Explain the following records with their formats:
      i) Header Record   ii) Text Record   iii) End Record   **(06 Marks)**
   c. What are the various data structures used by assembler? Explain.   **(06 Marks)**

OR

2 a. Write the Pass 1 algorithms for two pass assembler.   **(08 Marks)**
   b. List the various machine independent assembler features? Explain any one feature in detail.
   (06 Marks)
   c. What are the basic functions of the loader? Write an algorithm for design of an absolute loader.   **(06 Marks)**

Module-2

3 a. Explain the various phases of the compiler? Clearly specify the output at each phase for the input $A = B * C + 369$ **(10 Marks)**
   b. List and explain the reasons for separating analysis phase into lexical and syntax. **(04 Marks)**
   c. What are the applications of compiler technology? Discuss any two.   **(06 Marks)**

OR

4 a. Explain various input buffering schemes used in lexical analysis? Write the look ahead code for sentinel.   **(08 Marks)**
   b. Enlist the algebraic laws for regular expressions.   **(04 Marks)**
   c. Give the Regular definition and draw the transition diagram for
      i) Relational operator in C
      ii) Unsigned number
      iii) Indentifier and keyword   **(08 Marks)**

Module-3

5 a. Define ambiguity. Show that the grammar $E \rightarrow E + E \mid E * E \mid id$ is ambiguous? Eliminate the ambiguity and rewrite the grammar.   **(08 Marks)**
   b. For the following grammar eliminate the left recursion and for the resultant grammar construct the LL(1) parsing table and parse the input string (a, a)
      $$S \rightarrow (L) \mid a$$
      $$L \rightarrow L, S \mid S$$ **(08 Marks)**
   c. Give an algorithm for recursive descent parsing? What are its limitation and how to overcome it?   **(04 Marks)**

OR

6 a. Consider the following grammar

$$E \rightarrow 1 + T \mid 2 - T$$
$$T \rightarrow V \mid V * V \mid V + V \mid V - V$$
$$V \rightarrow a \mid b$$

i) Do the left factoring
ii) Write an algorithm for FIRST and follow and obtain it for the left factored grammar
iii) Construct it for the above left factored grammar. **(10 Marks)**

b. What is shift reduce parsing? Explain the conflicts that may occur during shift reduce parsing? Show the working of shift reduce parser for the following grammar and input string id * id

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid id$$

(10 Marks)

Module-4

7 a. What is lex? With an example explain the structure of lex program. **(06 Marks)**
b. Write the regular compression to identify the following :
i) Identifier   ii) Decimal number   iii) −ve integer   iv) +ve fraction **(08 Marks)**
c. Write a yacc program to evaluate an arithmetic expression. **(06 Marks)**

OR

8 a. Explain the yacc tool with a sample program. **(08 Marks)**
b. Write a short note on parser-lexer communication. **(06 Marks)**
c. Discuss how to compile a yacc file. **(06 Marks)**

Module-5

9 a. Give the SDD for a simple desk calculator and show the annotated parse tree for $(3 + 4) * (5 + 6)$ n **(08 Marks)**
b. Give the SDD for simple type declaration construct a dependency graph for the declaration
int sum, num1, num2; **(06 Marks)**
c. Explain how DAG helps in intermediate code generation? Construct a DAG for the following :
(i) $a + b + (a + b)$
(ii) $a + b + a + b$ **(06 Marks)**

OR

10 a. What are the different three address code instructions? Translate the arithmetic expression $a + (-(b + c))$ into quadruples, triples and indirect triples. **(08 Marks)**
b. Explain the issues in design of code generator. **(08 Marks)**
c. Generate the assembly code for the following address statements.
(i) $x = b * c$ (ii) $y = a + x$ **(04 Marks)**

* * * * *