Reg. No. : | | | | | | | | | | | | |

Question Paper Code : 20873

B.E./B.Tech. DEGREE EXAMINATIONS, NOVEMBER/DECEMBER 2023.

Fifth Semester

Computer Science and Engineering

CS 3501 — COMPILER DESIGN

(Regulations 2021)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Define compiler.
2. What is finite automata?
3. What are the goals of error handler in a parser?
4. Define a parse tree.
5. What is meant by Back patching?
6. What is meant by Type Checking?
7. Define symbol table.
8. What is a flow graph? Give example.
9. What is DAG? Give example.
10. What is basic block?

PART B — (5 × 13 = 65 marks)

11. (a) What are the phases of the compiler? Explain each phase in detail. (13)

Or

(b) Briefly discuss about Role of Lexical Analyzer. (13)

12. (a) Briefly discuss about Design of a syntax Analyzer for a sample language. (13)

Or

(b) Explain the LR parsing algorithm with an example. (13)

13. (a) Explain in detail about Design of predictive translator. (13)

Or

(b) Discuss the address code and its implementation, with example. (13)

14. (a) Discuss about the run time storage management of a code generator in detail. (13)

Or

(b) What are the issues in the design of the code generator? Explain in detail. (13)

15. (a) Explain the principle sources of code optimization in detail. (13)

Or

(b) Discuss the DAG representation of the basic block with an example. (13)

PART C — ($1 \times 15 = 15$ marks)

16. (a) How the context free grammar will work and compare various parser techniques performance using various code blocks? (15)

Or

(b) How the data flow optimization can be done and compare the features? (15)

Reg. No. : | | | | | | | | | | | | |

Question Paper Code : 50906

B.E./B.Tech. DEGREE EXAMINATIONS, APRIL/MAY 2024.

Fifth Semester

Computer Science and Engineering

CS 3501 — COMPILER DESIGN

(Regulations 2021)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. State the difference between a compiler and an interpreter.

2. Define the interaction between lexical analyzers and parsers.

3. What are the common programming errors that can occur at different levels?

4. What is the purpose of LR parsers?

5. Define inherited attribute.

6. State the rules of type checking.

7. What are two strategies for dynamic storage allocation?

8. Write the role of activation record.

9. What are the causes of redundancy?

10. What is copy propagation?

PART B — (5 × 13 = 65 marks)

11. (a) (i) Divide the following C++ program. (5)

```
float limitedSquare(x) {
/* returns x-squared, but never float x; more than 100*/
return (x<=–10.0| |x>=10.0)? 100:x*x;
}
```

Into appropriate lexemes. Which lexemes should get associated lexical values? What should those values be?

(ii) Explain the role of lexical analyzer. (8)

Or

(b) (i) Describe the languages denoted by the following regular expressions : (5)

(1) a(a | b)*a

(2) ((ε | a)b*)*

(3) (a | b)*a(a | b)(a | b)

(4) a*ba*ba*ba*

(5) !! (aa | bb)*((ab | ba) (aa | bb)*(ab | ba)(aa | bb)*)*.

(ii) Explain the construction of a DFA from a regular expression. (8)

12. (a) (i) Consider the context-free grammar.

S - > SS + | S S * | a

and the string aa + a*

(1) Give a leftmost derivation for the string. (1)

(2) Give a rightmost derivation for the string. (1)

(3) Give a parse tree for the string. (1)

(4) Is the grammar ambiguous or unambiguous? Justify your answer. (2)

(ii) Explain the error handling and error recovery mechanism of syntax analyser. (8)

Or

(b) (i) Show that the following grammar. (5)

S -> S A | A

A - > a

is SLR(1) but not LL(1).

(ii) State and explain the algorithm to eliminate left recursion from a grammar. (8)

13. (a) (i) The expressions involving operator + and integer or floating-point operands. Floating-point numbers are distinguished by having a decimal point. (5)

E -> E + T | T

T - > num.num | num

Give an SDD to determine the type of each term T and expression E.

(ii) Explain the S-Attribute Definitions with example. (8)

Or

(b) (i) Translate the arithmetic expression a + (b + c) into (5)

(1) Syntax tree

(2) Quadruples.

(ii) State and explain the translation of expressions and the addressing array elements with example. (8)

14. (a) What are the issues in the design of a code generator? Explain with example.

Or

(b) Explain the stack allocation of space with example? What principles are helpful when designing calling sequences?

15. (a) What are induction variable and how to find the induction variable in loops and optimize their computation. Explain with example.

Or

(b) Explain the optimization of basic blocks? How does the Directed Acyclic Graph (DAG) representation of a basic block facilitate code-improving transformations within the represented code?

PART C — (1 × 15 = 15 marks)

16. (a) Construct the DAG and identify the value numbers for the subexpressions of the following expressions, assuming + associates from the left.

(i) a + b + (a + b)

(ii) a + b + a + b

(iii) a + a + (a + a + a + (a + a + a + a)).

Or

(b) Explain the steps involved in the construction of predictive parser with suitable example.

———————