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

Question Paper Code : 50398

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2017

Sixth Semester

Computer Science and Engineering

CS6660 – COMPILER DESIGN

(Common to : Information Technology)

(Regulations 2013)

Time : Three Hours Maximum : 100 Marks

Answer ALL questions

PART – A (10×2=20 Marks)

1. What is an interpreter ?
2. What do you mean by Cross-Compiler ?
3. What is the role of lexical analysis phase ?
4. Define Lexeme.
5. Draw syntax tree for the expression a=b*– c+b*– c.
6. What are the three storage allocation strategies ?
7. Differentiate NFA and DFA.
8. Compare syntax tree and parse tree.
9. Draw the DAG for the statement a = (a*b+c)–(a*b+c).
10. What are the properties of optimizing compilers ?

PART – B (5×16=80 Marks)

11. a) What are compiler construction tools ? Write note on each Compiler Construction tool.

(OR)

b) Explain in detail the various phases of compilers with an example.

12. a) i) Discuss the issues involved in designing Lexical Analyzer.

ii) Draw NFA for the regular expression ab*/ab.

(OR)

b) Write an algorithm to convert NFA to DFA and minimize DFA. Give an example.

13. a) Explain LR parsing algorithm with an example.

(OR)

b) Explain the non-recursive implementation of predictive parsers with the help of the grammar.

E-> E+T | T

T->T*F | F

F->(E) | id

14. a) Explain the specification of simple type checker for statements, expressions and functions.

(OR)

b) Explain about runtime storage management.

15. a) Discuss the issues in code generation with examples.

(OR)

b) Explain briefly about the principal sources of optimization.

12. a) i) Discuss the issues involved in designing Lexical Analyzer.

ii) Draw NFA for the regular expression ab*/ab.

(OR)

b) Write an algorithm to convert NFA to DFA and minimize DFA. Give an example.

13. a) Explain LR parsing algorithm with an example.

(OR)

b) Explain the non-recursive implementation of predictive parsers with the help of the grammar.

E-> E+T | T

T->T*F | F

F->(E) | id

14. a) Explain the specification of simple type checker for statements, expressions and functions.

(OR)

b) Explain about runtime storage management.

15. a) Discuss the issues in code generation with examples.

(OR)

b) Explain briefly about the principal sources of optimization.