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

Question Paper Code : 57263

AN
08/6/16

B.E./ B.Tech. DEGREE EXAMINATION, MAY/JUNE 2016

Sixth Semester

Computer Science and Engineering

CS 6660– COMPILER DESIGN

(Regulations 2013)

Time : Three Hours **Maximum : 100 Marks**

Answer ALL questions.

PART – A (10 × 2 = 20 Marks)

1. What are the two parts of a compilation ? Explain briefly.
2. Illustrate diagrammatically how a language is processed.
3. Write a grammar for branching statements.
4. List the operations on languages.
5. Write the algorithm for FIRST and FOLLOW in parser.
6. Define ambiguous grammar.
7. What is DAG ?
8. When does Dangling references occur ?
9. What are the properties of optimizing compiler ?
10. Write three address code sequence for the assignment statement

    d:= (a-b) + (a-c) + (a-c).

PART – B (5 × 16 = 80 Marks)

11. (a) Describe the various phases of compiler and trace it with the program segment (position:= initial + rate * 60). **(16)**

OR

(b) (i) Explain language processing system with neat diagram. **(8)**

(ii) Explain the need for grouping of phases. **(4)**

(iii) Explain various Error encountered in different phases of compiler. **(4)**

12. (a) (i) Differentiate between lexeme, token and pattern. **(6)**

(ii) What are the issues in lexical analysis ? **(4)**

(iii) Write notes on regular expressions. **(6)**

OR

(b) (i) Write notes on regular expression to NFA. Construct Regular expression to NFA for the sentence (a|b)* a. **(10)**

(ii) Construct DFA to recognize the language (a/b)* ab. **(6)**

13. (a) (i) Construct Sack implementation of shift reduce parsing for the grammar **(8)**

E -> E+E
E -> E*E
E -> (E)
E -> id and the input string id1 + id2 *id3

(ii) Explain LL(1) grammar for the sentence S->iEts | iEtSeS | a E->b. **(8)**

OR

(b) (i) Write an algorithm for Non recursive predictive parsing. **(6)**

(ii) Explain Context free grammers with examples. **(10)**

14. (a) (i) Construct a syntax directed definition for constructing a syntax tree for assignment statements. **(8)**

S → **id** : = E
E → E1 + E2
E →E1 * E2
E →-E1
E →(E1)
E →**id**

(ii) Discuss specification of a simple type checker. **(8)**

OR

(b) Discuss different storage allocation strategies. **(16)**

15. (a) Explain Principal sources of optimization with examples. **(16)**

OR

(b) (i) Explain various issues in the design of code generator. **(8)**

(ii) Write note on simple code generator. **(8)**

———————

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

Question Paper Code : 71689

B.E./B.Tech. DEGREE EXAMINATION, APRIL/MAY 2017.

Sixth Semester

Computer Science and Engineering

CS 6660 — COMPILER DESIGN

(Common to Information Technology)

(Regulations 2013)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Define the two parts of compilation.

2. List the cousins of the compiler?

3. Write a regular expression for an identifier and number.

4. What are the various parts in LEX program?

5. Eliminate the left recursion for the grammar A→ Ac | Aad | bd.

6. What are the various conflicts that occur during shift reduce parsing?

7. What do you mean by binding of names?

8. Mention the rules for type checking.

9. What is a basic block?

10. What do you mean by copy propagation?

PART B — (5 × 16 = 80 marks)

11. (a) What are the phases of the compiler? Explain the phases in detail. Write down the output of each phase for the expression a : = b + c *60. (16)

Or

(b) (i) Explain briefly about compiler construction tools. (6)

(ii) Describe in detail about Cousins of compiler? (4)

(iii) Draw the transition diagram for relational operators and unsigned numbers. (6)

12. (a) Convert the Regular Expression abb (a | b)* to DFA using direct method and minimize it. (16)

Or

(b) (i) Differentiate between lexeme, token and pattern. (6)

(ii) What are the issues in lexical analysis? (4)

(iii) Draw the transition diagram for relational operators and unsigned numbers. (6)

13. (a) Construct a predictive parsing table for the grammar

$S \rightarrow (L) \mid a$

$L \rightarrow L, S \mid S.$

and show whether the following string will be accepted or not. (a,(a,(a,a))). (16)

Or

(b) Consider the following Grammar

E → E+T | T

T→ TF | F

F→ F* | a | b

Construct the SLR parsing table for the above grammar. (16)

14. (a) What are the different storage allocation strategies? (16)

Or

(b) (i) Explain in detail about Specification of a simple type checker (10)

(ii) Explain about the parameter passing. (6)

15. (a) Discuss the various issues in design of Code Generator. (16)

Or

(b) (i) Explain in detail about optimization of Basic Blocks. (8)

(ii) Construct the DAG for the following Basic Block. (8)

1. t1 : = 4*i
2. t2 : = a[t1]
3. t3 : = 4*i
4. t4 : = b[t3]
5. t5 : = t2*t4
6. t6 : = prod+t5
7. prod : = t6
8. t7 : = i+1
9. i : = t7
10. if i < = 20 goto (1).

———————

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

Question Paper Code : 80303

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2016.

Sixth Semester

Computer Science and Engineering

CS 6660 — COMPILER DESIGN

(Common to Sixth Semester Information Technology)

(Regulations 2013)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. What is a symbol table?
2. List the various compiler construction tools.
3. List the rules that form the BASIS.
4. Differentiate tokens, patterns, lexeme.
5. Construct a parse tree for –(id + id)
6. What is meant by handle pruning?
7. Write down syntax directed definition of a simple desk calculator.
8. List Dynamic Storage allocation techniques.
9. Identify the constructs for optimization in basic block.
10. What are the characteristics of peephole optimization?

PART B — (5 × 16 = 80 marks)

11. (a) (i) Explain the phases of compiler with a neat diagram. (10)

(ii) Write notes on compiler Construction tools. (6)

Or

(b) (i) Explain the need for grouping of phases. (8)

(ii) Explain the various errors encountered in different phases of compiler. (8)

12. (a) (i) Discuss the role of lexical analyzer in detail with necessary examples. (8)

(ii) Discuss how finite automata is used to represent tokens and perform lexical analysis with examples. (8)

Or

(b) (i) Conversion of regular expression (a/b)*abb to NFA. (8)

(ii) Write an algorithm for minimizing the number of states of a DFA. (8)

13. (a) (i) Construct parse tree for the input string w = cad using top down parser. (6)

$S \rightarrow cAd$

$A \rightarrow ab \mid a$

(ii) Construct parsing table for the grammar and find moves made by predictive parser on input id+id*id and find FIRST and FOLLOW. (10)

$E \rightarrow E + T$

$E \rightarrow T$

$T \rightarrow T * F$

$T \rightarrow F$

$F \rightarrow (E)/id$

Or

(b) (i) Explain ambiguous grammar $G : E \rightarrow E + E \mid E * E \mid (E) \mid -E \mid id$ for the sentence id+id*id. (6)

(ii) Construct SLR parsing table for the following grammar :

$G : E \rightarrow E + T \mid TT \rightarrow T * F \mid FF \rightarrow (E) \mid id$. (10)

14. (a) (i) A Syntax-Directed Translation scheme that takes strings of a's, b's and c's as input and produces as output the number of substrings in the input string that correspond to the pattern a(a | b)*c+(a | b)*b. For example the translation of the input string "abbcabcababc" is "3".

(1) Write a context-free grammar that generate all strings of a's. b's and c's.

(2) Give the semantic attributes for the grammar symbols.

(3) For each production of the grammar present a set of rules for evaluation of the semantic attributes. (8)

(ii) Illustrate type checking with necessary diagram. (8)

Or

(b) Explain the following with respect to code generation phase.

(i) Input to code generator

(ii) Target program

(iii) Memory management

(iv) Instruction selection

(v) Register allocation

(vi) Evaluation order. (16)

15. (a) (i) Write an algorithm for constructing natural loop of a back edge. (8)

(ii) Explain any four issues that crop up when designing a code generator. (8)

Or

(b) Explain global data flow analysis with necessary equations. (16)

———————

Reg. No. :

04/12/17 (AN)

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.

13/12/18

Reg. No. :

Question Paper Code : 20374

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2018.

Sixth Semester

Computer Science and Engineering

CS 6660 — COMPILER DESIGN

(Common to Information Technology)

(Regulations 2013)

(Also common to PTCS 6660 — Compiler Design – for B.E. (Part-Time) Fifth Semester – Computer Science and Engineering – Regulations 2014)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Recall the basic the two parts of a compilation process.
2. How a source code is translated to machine code?
3. State the rules to define regular expression.
4. Construct Regular expression for the language $L = \{w \in \{a, b\} / w \text{ ends in } abb\}$.
5. What are the different stages that a parser can recover from a syntactic error?
6. Define LR (0) item.
7. List three kinds of intermediate representation.
8. When procedure call occurs, what are the steps taken?
9. State the problems in code generation.
10. Define common sub expression.

PART B — (5 × 13 = 65 marks)

11. (a) Write short notes about :

(i) Compiler Construction Tools. (7)

(ii) Lexeme, token and pattern. (6)

Or

(b) Discuss in detail about the operations of compiler which transforms the source program from one representation into another. Illustrate the output for the input : (13)

$a = (b + c) * (b + c) * 2$.

12. (a) Write briefly about :

(i) the role of Lexical analyzer with the possible error Recovery actions. (5)

(ii) recognition and specification of tokens. (8)

Or

(b) Construct the minimized DFA for the regular expression $(0+1)*(0+1)01$. (13)

13. (a) Show that the following grammar

$S \rightarrow Aa \mid bAc \mid dc \mid bda$

$A \rightarrow a$

is LALR(1) but not SLR(1). (13)

Or

(b) Show that the following grammar

$S \rightarrow Aa \mid bAc \mid Bc \mid bBa$

$A \rightarrow d$

$B \rightarrow d$

is LR(1) but not LALR(1). (13)

14. (a) Apply the S-attributed definition and constructs syntax trees for a simple expression grammar involving only the binary operators + and –. As usual, these operators are at the same precedence level and are jointly left associative. All nonterminal have one synthesized attribute node, which represents a node of the syntax tree.

Production: $E \rightarrow E_1 + T$, $E \rightarrow T$, $T \rightarrow (E)$, $T \rightarrow$ id/num . (13)

Or

(b) Discuss in detail about :

(i) Storage allocation strategies. (7)

(ii) Parameter passing methods. (6)

15. (a) Discuss in detail about optimization of basic blocks. (13)

Or

(b) Explain in detail about issues in the design of a code generator. (13)

PART C — (1 × 15 = 15 marks)

16. (a) Suppose we have a production $A \rightarrow B\ C\ D$. Each of the four nonterminals has two attributes s, which is synthesized, and i, which is inherited. For each set of rules below, check whether the rules are consistent with (i) an S-attributed definition, (ii) an L-attributed definition (iii) any evaluation order at all.

(1) $A.s = B.i + C.i$

(2) $A.s = B.i + C.s$ and $D.i = A.i + B.s$

(3) $A.s = B.s + D.s$

(4) $A.s = D.i$
$B.i = A.s + C.s$
$C.i = B.s$
$D.i = B.i + C.i$. (15)

Or

(b) Construct a Syntax-Directed Translation scheme that translates arithmetic expression from infix into postfix notation. (15)

15/11/19/AN

Reg. No. :

b) Consider the code

```
01          a = 1
02          b = 2
03 L0:      c = a + b
04          d = c – a
05          if c < d goto L2
06L1:       d = b + d
07          if d < 1 goto L3
08L2:       b = a + b
09          e = c – a
10          if e = 0 goto L0
11          a = b + d
12          b = a – d
13          goto L4
14L3:       d = a + b
15          e = e + 1
16          goto L3
17L4:       return
```

For the code shown above, determine the following :

i) The basic blocks of instructions **(3)**

ii) The control-flow graph (CFG) **(3)**

iii) For each variable, its corresponding def-use chain **(3)**

iv) The live variables at the end of each basic block. You do not need to determine the live variables before and after each basic block and justify your answer for the value presented for the basic block containing instructions at line 6 and 7. **(3)**

v) Is the live variable analysis a forward or backward data-flow analysis problem ? Why and what does guarantee its termination when formulated as a data-flow analysis iterative problem ? **(3)**

Question Paper Code : 91408

B.E./B.Tech. DEGREE EXAMINATIONS, NOVEMBER/DECEMBER 2019

Sixth Semester

Computer Science and Engineering

CS 6660 – COMPILER DESIGN

(Common to Information Technology)

(Regulations 2013)

(Also common to PTCS 6660 – Compiler Design for B.E. (Part-Time) Fifth Semester – Computer Science and Engineering – Regulations 2014)

Time : Three Hours Maximum : 100 Marks

Answer ALL questions

PART – A **(10×2=20 Marks)**

1. Name a compiler construction tool used to design i) Lexical Analyser and ii) Parser.

2. Which phase (or phases) of a compiler
   i) is/are considered the "back end" ?
   ii) access (es) the symbol table (for reading or writing) ?
   iii) check (s) for type mismatches ?
   iv) is/are independent of the underlying machine ?

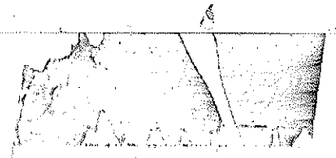
3. Consider the language of all strings from the alphabet {a, b, c} containing the substring "abcabb". Write a regular expression that describes this language.

4. Give any two reasons for keeping lexical analyser a separate phase instead of making it an integral part of syntax analysis.

5. Show that the grammar, $S \rightarrow aSbS \mid bSaS \mid \varepsilon$ is ambiguous.

6. Give the structure of YACC program to design a simple syntax analyser.

7. Differentiate implicit and explicit type conversions.



8. What is an activation record ? Give the structure of an activation record.

9. Construct a DAG for the following code

   a = b + c

   b = a – d

   c = b + c

   d = a – d

10. What is the cost of the following sequences of instructions ?

    i) MOV b, a
       ADD c, a **(1)**

    ii) MOV * R1, *R0
        ADD * R2, *R0 **(1)**

PART – B **(5×13=65 Marks)**

11. a) What are the different phases of a compiler ? Write their functions. Show how the high level language statement position = initial + rate * 60 is converted to machine code by each phase.

(OR)

b) What are the components of a language processing system ? Explain the role of each of these components in a typical compilation and execution of the program.

12. a) Given the regular expression (a | b)*abb over the alphabet Σ = {a, b}

    i) Construct a NFA with ε-transitions using Thompson Construction. **(4)**

    ii) Convert the NFA obtained from (i) to non-minimal DFA. **(5)**

    iii) Minimize the number of states obtained from ii) to minimal DFA. **(4)**

(OR)

b) Write a lex program to implement a calculator. Describe the actions of the program and the functions defined and used.

13. a) Consider the following grammar G :

    $S' \rightarrow S$

    $S \rightarrow CC$

    $C \rightarrow cC \mid d$

    Construct the LALR parsing table for the grammar G. Show the moves of the parser on the string ccd

(OR)

b) Explain the construction of predictive parsing table and describe the moves of the parser on an input string. Design a predictive parser for the following grammar

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid id$

Show the moves of the parser on the input id + id * id.

14. a) What is a syntax tree ? Describe construction of syntax trees for expressions ? Give examples to support your description. Write syntax directed definition for constructing syntax trees/Draw an annotated parse tree for the expression a – 4 + c.

(OR)

b) What is a symbol table ? What type of information is stored in it ? Discuss on the use of the data structures i. arrays ii. Linked lists iii. Binary search trees for implementing a symbol table.

15. a) What are the principal sources of optimization ? Explain with suitable examples.

(OR)

b) Write a simple code generator algorithm. With an example code, show how the algorithm generates code.

PART – C **(1×15=15 Marks)**

16. a) Consider the following grammar G :

    $S \rightarrow XaY \mid Y$

    $X \rightarrow bY \mid c$

    $Y \rightarrow X$

    i) Discuss the various steps involved in the construction of SLR parsing. **(3)**

    ii) Show the canonical collection of LR(0) items. **(5)**

    iii) Construct the SLR parsing table. **(4)**

    iv) Show the action of the parser on the input string cac$. **(3)**

(OR)