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

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)

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)