Reg. No. : □□□□□□□□□□□□

Question Paper Code : X 10323

B.E./B.Tech. DEGREE EXAMINATIONS, NOVEMBER/DECEMBER 2020
Sixth Semester
Computer Science and Engineering
CS8602 – COMPILER DESIGN
(Regulations 2017)

Time : Three Hours | Maximum : 100 Marks

Answer ALL questions

PART – A **(10×2=20 Marks)**

1. What advantages are there to a language-processing system in which the compiler produces assembly language rather than machine language ?

2. With a neat block diagram specify the interactions between the lexical analyzer and the parser.

3. State the various error recovery strategies used in a parser to correct the errors.

4. What is bottom up parsing and shift reduce parsing ?

5. What are Inherited and Synthesized attributes ?

6. 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) $a*b*a*b$

7. Differentiate between static and dynamic storage allocation.

8. State the tasks of a code generator.

9. Brief about the methodology used to locally improve the target code.

10. What is a basic block ? Give an example.

PART – B **(5×13=65 Marks)**

11. a) What are Lexical errors ? What are the possible recovery mechanisms ? Divide the following C++ program :

```
float limited Square(x) float x;
/* returns x-squared, but never 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 ? **(13)**

(OR)

b) What is a transition diagrams ? Explain briefly how the keywords and identifiers are recognized using a running example. **(13)**

12. a) A grammar symbol X (terminal or nonterminal) is useless if there is no derivation of the form S wXy wxy That is, X can never appear in the derivation of any sentence. Elaborate on the algorithm that is used to eliminate from a grammar all productions containing useless symbols. Apply your algorithm to the grammar :

$S \rightarrow 0 \mid A$

$A \rightarrow AB$

$B \rightarrow 1$ **(13)**

(OR)

b) Consider the following grammar and construct SLR parser.

$E \rightarrow E + T/T,\ T \rightarrow T * F \mid F,\ F \rightarrow (E)\ \mid id.$ **(13)**

13. a) Describe how SDD can be evaluated at the nodes of a parse tree using dependency graphs. **(13)**

(OR)

b) Explain type checking and type conversion. Explain with an example of converting the operands the same type. **(13)**

14. a) What is the Memory Hierarchy configuration of a computer ? Discuss the memory manager subsystem that is responsible for allocating and deallocating space within the heap. **(13)**

(OR)

b) Illustrate the algorithm that generates code for a single basic block with three address instructions. **(13)**

15. a) What is code optimization ? State its advantages. Discuss various code optimization schemes in detail. **(13)**

(OR)

b) Discuss about the following with example: **(13)**

i) Copy Propagation

ii) Dead-code Elimination and

iii) Code motion.

PART – C **(1×15=15 Marks)**

16. a) Consider the following CFG

E→E or T | T

T→T and F | F

F→not F | (E) | true | false

Write the semantic rules and explain the processes converting "not (true or false)" to intermediate form using Parser tree method. **(15)**

(OR)

b) Consider the grammar S → ABD, A → a | Db | ε, B → gD | dA | ε , D → e | f

i) Construct FIRST and FOLLOW for each nonterminal of the above grammar.

ii) Construct the predictive parsing table for the grammar.

iii) Show the parsing action on a valid string and on an invalid string

iv) Check whether the grammar is LL (1). Give justification. **(15)**