# INSTITUTEOFAERONAUTICALENGINEERING

**(Autonomous)**

Dundigal, Hyderabad-500043

## COMPUTER SCIENCE AND ENGINEERING

## TUTORIAL QUESTION BANK

| | | | | | |
|---|---|---|---|---|---|
| **Course Title** | **COMPILER DESIGN** | | | | |
| **Course Code** | ACSB11 | | | | |
| **Programme** | B.Tech | | | | |
| **Semester** | V | CSE \| IT | | | |
| **Course Type** | Core | | | | |
| **Regulation** | **IARE - R18** | | | | |
| **Course Structure** | **Theory** | | | **Practical** | |
| | **Lectures** | **Tutorials** | **Credits** | **Laboratory** | **Credits** |
| | 2 | 1 | 3 | - | - |
| **Chief Coordinator** | Mr. N V Krishna Rao, Assistant Professor | | | | |

## COURSE OBJECTIVES:

**The students will try to learn:**

| | |
|---|---|
| I | The process of translating a high-level language to machine code required for compiler construction. |
| II | The Software tools and techniques used in compiler construction such as lexical analyser and parser generators. |
| III | The data structures used in compiler construction such as abstract syntax trees, symbol tables, three-address code, and stack machines. |
| IV | The deeper insights into the syntax and semantic aspects of programming languages, dynamic memory allocation and code generation. |

## COURSE OUTCOMES :

**After successful completion of the course, Students will be able to:**

| | |
|---|---|
| CO 1 | **Describe the components of a language processing system for the conversion of high level languages to machine level languages.** |
| CO 2 | **Classify the importance of phases of a compiler for constructing a compiler** |
| CO 3 | **Demonstrate a lexical analyser from a specification of a Language's lexical rules for dividing the programming statements into tokens.** |
| CO 4 | **Construct the derivations , FIRST set , FOLLOW set on the context free grammar for performing the top-down and bottom up parsing methods.** |
| CO 5 | **Distinguish top down and bottom up parsing methods for developing parser with the parse tree representation of the input.** |
| CO 6 | **Construct LEX and YACC tools for developing a scanner and a parser.** |
| CO 7 | **Describe syntax directed definitions & translations for performing Sementic Analysis.** |
| CO 8 | **Classify the different intermediate forms for conversion of syntax translations into Intermediate Code.** |
| CO 9 | **Demonstrate type systems for performing the static and dynamic type checking.** |

| | |
|---|---|
| CO 10 | Describe the run-time memory elements for storage allocation strategies which includes procedure calls, local variable allocation, dynamic memory allocation. |
| CO 11 | Apply the code optimization techniques on intermediate code form for improving the performance of a program |
| CO 12 | Make use of optimization techniques on basic blocks for reducing utilization of registers in generating the target code. |

## TOTAL COUNT OF KEY COMPETENCIES FOR CO – (PO, PSO) MAPPING:

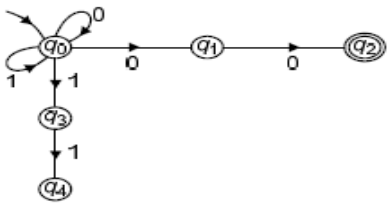
| Course Outcomes(COs) | Program Outcomes (POs) / Number of Vital Features | | | | | | | | | | | | Program Specific Outcomes (PSOs) / Number of Vital Features | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 1 | 2 | 3 |
| | 3 | 10 | 10 | 11 | 1 | 5 | 3 | 3 | 12 | 5 | 12 | 12 | 1 | 2 | 2 |
| CO 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | 1 | - | - |
| CO 2 | 2 | - | - | - | 1 | - | - | - | - | - | - | - | 1 | - | - |
| CO 3 | 3 | - | - | - | - | - | - | - | - | - | - | - | 1 | - | - |
| CO 4 | 2 | 2 | - | - | - | - | - | - | - | - | - | - | - | 1 | - |
| CO 5 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | 2 | - |
| CO 6 | 2 | 2 | - | 1 | - | - | - | - | - | - | - | - | - | 2 | - |
| CO 7 | 2 | - | 2 | - | - | - | - | - | - | - | - | - | - | - | - |
| CO 8 | 2 | - | 2 | - | 1 | - | - | - | - | - | - | - | - | - | - |
| CO 9 | 2 | - | 1 | - | - | - | - | - | - | - | - | - | - | - | - |
| CO 10 | - | 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - |
| CO 11 | 2 | 3 | | - | 1 | - | - | - | - | 1 | - | - | - | - | 1 |
| CO 12 | - | 2 | 2 | - | 1 | - | - | - | - | 1 | - | - | - | - | 1 |

# TUTORIAL QUESTION BANK

| MODULE-I | | | | |
|---|---|---|---|---|
| INTRODUCTION TO COMPILERS | | | | |
| PART - A (SHORT ANSWER QUESTIONS) | | | | |
| S No | QUESTIONS | Blooms Taxonomy Level | | Course Outcomes |
| 1 | Name the cousins of compiler? | Remember | --- | CO 1 |
| 2 | Define the two main parts of compilation? What they perform? | Remember | --- | CO 1 |
| 3 | How many phases does analysis phase consists define it? | Remember | --- | CO 2 |
| 4 | Define and explain the Loader? | Remember | --- | CO 1 |

| | | | | |
|---|---|---|---|---|
| 5 | Define preprocessor? | Remember | --- | CO 1 |
| 6 | Define the general phases of a compiler? | Remember | --- | CO 2 |
| 7 | Define lexeme and token? | Remember | --- | CO 3 |
| 8 | Write the issues of lexical analyzer? | Remember | --- | CO 3 |
| 9 | List some compiler construction tools? | Remember | --- | CO 1 |
| 10 | Define the term Symbol table? | Remember | --- | CO 2 |
| 11 | Define the term Interpreter? | Remember | --- | CO 1 |
| 12 | Define an error Handler in compiler? | Remember | --- | CO 2 |
| 13 | Explain a translator and types of translator? | Understand | **Recall** components of a language processing system for the **convert** high level languages to machine level languages. | CO 1 |
| 14 | Define parser and list its types? | Remember | --- | CO 2 |
| 15 | Explain bootstrap? | Understand | This would require the learner to **recall** (knowledge) basic components of a language processing system to **construct** tokens, lexemes, different symbols and their importance and applicability in implementing the lexical analyser | CO 1 |
| 16 | Define pass? | Remember | --- | CO 2 |
| 17 | Define phase? | Remember | --- | CO 2 |
| 18 | What is cross compiler? | Remember | --- | CO1 |
| 19 | Define multi pass compiler? | Remember | --- | CO 2 |
| 20 | Define DFA,NFA,Regular Expressions. | Remember | --- | CO 3 |
| **PART - B (LONG ANSWER QUESTIONS)** | | | | |
| 1 | Define compiler? State various phases of a compiler and explain them in detail? | Remember | --- | CO 2 |
| 2 | Explain the various phases of a compiler in detail. Also Write down the output for the following expression after each phase x: =a+b*c-d? | Understand | This would require the learner to **recall** the various phases of compiler and **discuss** the output of each phase for given expression. | CO2 |
| 3 | Explain the cousins of a Compiler? Explain them in detail. | Understand | This would require the learner to **recall** the different phases of compiler, **classify** the cousins of compiler | CO 1 |
| 4 | Describe how various phases could be combined as a pass in compiler? | Remember | --- | CO 2 |
| 5 | Convert Regular Expression (11+0)*(00+1)* to Finite Automata. | Understand | This would require the learner to **recall** the finite automata and **show** the steps for the conversion of NFA to DFA. | CO 3 |
| 6 | For the following expression Position:=initial+ rate*60, Show the output after each phase of | Understand | This would require the learner to **recall** the various phases of compiler and **demonstrate** the | CO 2 |

| | | | | |
|---|---|---|---|---|
| | compiler? | | output of each phase for given expression. | |
| 7 | Explain the role and issues of Lexical Analyzer? | Understand | This would require the learner to **recall** the different phases of compiler, **classify** the cousins of compiler | CO 3 |
| 8 | Define Regular Expression and its properties. Give examples for Regular Expressions for given Finite Automatas. | Remember | --- | CO 3 |
| 9 | Explain single pass and multi pass compiler with example? | Understand | This would require the learner to **recall** the different phases of compiler **classify** the cousins of compiler | CO 2 |
| 10 | Define bootstrapping concept in brief? | Remember | --- | CO 1 |
| 11 | Explain the general format of a LEX program with example? | Understand | This would require the learner to **recall** software tools such LEX to develop a complete compiler and explain with an example. | CO 3 |
| 12 | Explain and differentiate frontend and backend of a compiler | Understand | This would require the learner to **recall** the different phases of compiler and **discuss** frontend and backend. | CO 2 |
| 13 | For the following expression a[index]=4+2 , Explain output after each phase of compiler? | Understand | This would require the learner to **recall** the various phases of compiler and **demonstrate** the output of each phase for given expression. | CO 2 |
| 14 | Convert NFA for (0 + 1)*(00 + 11)(0 + 1)* and Convert to DFA. | Understand | This would require the learner to **recall** the finite automata and **show** the steps for the conversion of NFA to DFA. | CO 3 |
| 15 | Convert Regular Expression 01* + 1 to Finite Automata. | Understand | This would require the learner to **recall** the finite automata, regular expressions and **show** the steps for the conversion of RE to FA. | CO 3 |
| 16 | Compare compiler and interpreter. | Understand | **Recall** components of a language processing system for the **convert** high level languages to machine level languages. | CO 1 |
| 17 | Explain the properties of strings and languages. | Understand | This would require the learner to **recall** the finite automata and **write** about strings and languages. | CO 3 |
| 18 | Compare lexical analyzer and syntax analyzer. | Understand | This would require the learner to **recall** the various phases of compiler and **discuss** the lexical and syntax analyzers. | CO 2 |
| 19 | Explain the reasons for separating scanner from parser. | Understand | This would require the learner to **recall** the various phases of compiler and **discuss** the scanner, parser | CO 2 |
| 20 | Compare the pass and phase in compiler construction? | Understand | This would require the learner to **recall** the different phases of compiler **classify** the cousins of compiler | CO 2 |

| PART - C (PROBLEM SOLVING AND CRITICAL THINKING QUESTIONS) | | | | |
|---|---|---|---|---|
| 1 | Consider the following fragment of C code:<br>float i, j;<br>i = i*70+j+2;<br>Construct the output at all phases of the compiler for above „C" code? | Apply | This would require the learner to **recall** the various phases of compiler and **explain** the task of each phase and **apply** the concepts for the conversion of the input of each phase for given expression. | CO 2 |
| 2 | Describe the languages denoted by the following regular expressions.<br>i. (0+1)*0(0+1)(0+1)<br>ii. 0*10*10*10* | Understand | This would require the learner to **recall** regular expressions and **explain** the languages for given expression. | CO 3 |
| 3 | Explain how LEX program perform lexical analysis to identify Identifiers, Comments, Numerical constants, Keywords, Arithmetic operators? | Understand | This would require the learner to **recall** software tool such LEX to **demonstrate** a complete compiler. | CO 3 |
| 4 | For the following expression total = count + rate * 5<br>Construct the output after each phase of compiler? | Apply | This would require the learner to **recall** the various phases of compiler and **explain** the task of each phase and **apply** the concepts for the conversion of the input of each phase for given expression. | CO 2 |
| 5 | Convert Regular Expression (b+aa)*a* to Finite Automata. | Understand | This would require the learner to **recall** the finite automata, regular expressions and **show** the steps for the conversion of RE to FA. | CO 3 |
| 6 | Explain the DFA that will accept those words from ∑= {a, b} where the number of a's is divisible by two and the number of b's is divisible by three. Sketch the transition table of the finite automata | Understand | This would require the learner to **recall** the finite automata and **show** the steps for the writing DFA. | CO 3 |
| 7 | Convert Regular Expression (11+0)*(00+1)* to NFA. | Understand | This would require the learner to **recall** the finite automata, regular expressions and **show** the steps for the conversion of RE to FA. | CO 3 |
| 8 | Convert the following NFA to DFA, as shown in fig. below<br>  | Understand | This would require the learner to **recall** the finite automata and **show** the steps for the conversion of NFA to DFA. | CO 3 |
| 9 | Convert NFA with ε to NFA for the following regular expression a*b*. | Understand | This would require the learner to **recall** the finite automata and **show** the steps for the conversion of NFA with ε to NFA. | CO 3 |
| 10 | Explain Regular Expressions and outline the transition diagrams for different programming constructs like identifier, number, relation operators. . | Understand | This would require the learner to **recall** the finite automata, regular Expressions, recognition of tokens and **write** the RE, transition diagrams for programming constructs. | CO 3 |

| MODULE-II | | | | |
|---|---|---|---|---|
| SYNTAX ANALYSIS | | | | |
| PART - A (SHORT ANSWER QUESTIONS) | | | | |
| 1 | Define about FIRST and state its rules? | Remember | --- | CO 4 |
| 2 | Define about FOLLOW and state its rules? | Remember | --- | CO 4 |
| 3 | Define LR(0) items in bottom up parsing? | Remember | --- | CO 5 |
| 4 | What LR(k) parsing stands for? | Remember | --- | CO 5 |
| 5 | List types of bottom up parsing techniques? | Remember | --- | CO 5 |
| 6 | Define goto function and closure function in LR parser? | Remember | --- | CO 5 |
| 7 | Why SLR and LALR are more economical to construct Canonical LR? | Remember | --- | CO 5 |
| 8 | Tell about handle pruning? | Remember | --- | CO 5 |
| 9 | What are error recovery types? | Remember | --- | CO 5 |
| 10 | List down the conflicts during shift-reduce parsing. | Remember | --- | CO 5 |
| 11 | List out the types LR(0) and LR(1) parsers? | Remember | --- | CO 5 |
| 12 | Describe about shift reduce parsing? | Understand | This would require the learner to **recall** bottom up parsing technique and **explain** basic idea of shift reduce parsing. | CO 5 |
| 13 | Define YACC parser? | Remember | --- | CO 6 |
| 14 | Compare CLR and LALR? | Understand | This would require the learner to **recall** bottom up parsing technique and **compare** between two bottom up parsing techniques CLR and LALR | CO 5 |
| 15 | Define an augmented grammar? | Remember | --- | CO 4 |
| 16 | Define shift action? | Remember | --- | CO 5 |
| 17 | Define Reduce action? | Remember | --- | CO 5 |
| 18 | Is left recursion elimination is required in bottom up parsing ?justify. | Understand | This would require the learner to **recall** bottom up parsing technique and **explain** the left recursion and then justify whether left recursion elimination is required for bottom up parsing or not. | CO 5 |
| 19 | List out differences between LL and LR parsers? | Remember | --- | CO 5 |
| 20 | List out the operations of shift reduce parsing? | Remember | --- | CO 5 |
| PART - B (LONG ANSWER QUESTIONS) | | | | |
| 1 | List the FIRST and FOLLOW sets for following grammar?<br>S → ACB / CbB / Ba<br>A → da / BC | Remember | --- | CO 4 |

| | | | | |
|---|---|---|---|---|
| | B → g / ∈<br>C → h / ∈ | | | |
| 2 | Explain the common conflicts that can be encountered in a shift-reduce parser? | Understand | This would require the learner to **recall** bottom up parsing technique and **explain** the common conflicts encountered in shift reduce parser and also explain the conflicts with suitable grammar. | CO 5 |
| 3 | Explain handle pruning in detail with example? | Understand | This would require the learner to **recall** bottom up parsing technique and **explain** the term handle pruning with example grammar | CO 5 |
| 4 | Consider the grammar<br>E → E + E \| E *E \| (E) \| id<br>Show the sequence of moves made by the shift-reduce parser on the input (id1+id2)*id3 and determine whether the given string is accepted by the parser or not? | Understand | This would require the learner to **recall** bottom up parsing technique and **explain** the acceptance of the string. | CO 5 |
| 5 | Explain the role of stack in shift reduce parsing method? | Understand | This would require the learner to **recall** bottom up parsing technique and **explain** role of stake in shift reduce parsing method. | CO 5 |
| 6 | Explain YACC-automatic parser generator. | Understand | This would require the learner to **recall** context free grammar and **explain** how CFG is represented in YACC | CO 6 |
| 7 | State the difference between SLR,CLR and LALR parsers in detail? | Remember | --- | CO 5 |
| 8 | Explain briefly about panic mode and phrase level error recovery techniques? | Remember | --- | CO 5 |
| 9 | Explain how to handle the error in ambiguous grammar with example? | Understand | This would require the learner to **recall** context free grammar and **explain** how to handle error in ambiguous grammar with example grammar. | CO 4 |
| 10 | Outline the LR Parsing model and write the LR parsing algorithm for constructing the parsing table? | Understand | This would require the learner to **recall** bottom up parsing technique and **explain** the components in the LR parsing diagram. | CO 5 |
| 11 | Consider the grammar,<br>P→ E<br>E → E+T<br>E→T<br>T→ id(E) T→id<br>And, state whether the following grammar is LR(0) or not? | Remember | --- | CO 5 |
| 12 | Write shift reduce parsing algorithm and show shift and reduce moves with an example? | Understand | This would require the learner to **recall** bottom up parsing technique and **explain** the shift reduce parsing method. | CO 5 |
| 13 | Explain the following terms<br>i) Canonical collection of items | Understand | This would require the learner to **recall** bottom up parsing | CO 5 |

| | | | | |
|---|---|---|---|---|
| | ii) Augmented Grammar<br>iii) Closure and go to Operation | | technique and **explain** operations performed on LR parsing techniques such as Augmented grammar, closure and goto operations along with LR(0) items. | |
| 14 | Consider the grammar<br>P→ E<br>E → E+T<br>E→ T T→id (E) T→ id<br>And, State whether the following grammar is SLR(1) or not? | Remember | --- | CO 5 |
| 15 | Outline the CLR Parsing model and write the CLR parsing algorithm for constructing the parsing table? | Understand | This would require the learner to **recall** bottom up parsing technique and **explain** the steps in algorithm for the construction of CLR parsing table | CO 5 |
| 16 | Explain the SLR(1) parsing table for the following grammar S→ Aa \| bAc\|dc\|bd<br>A→d | Understand | This would require the learner to **recall** bottom up parsing technique and **explain** the steps in algorithm for the construction of SLR parsing table | CO 5 |
| 17 | Compare LR parsers in detail? | Understand | This would require the learner to **recall** bottom up parsing technique and **compare** the LR parsing methods. | CO 5 |
| 18 | Consider the grammar S→ AS\| b<br>A → SA \| a<br>Explain the collection of sets of LR(0) items for this grammar? | Understand | This would require the learner to **recall** bottom up parsing technique and **explain** the steps in algorithm for writing the LR items | CO 5 |
| 19 | Show that the following grammar<br>S→ AaAb \| BbBa<br>A→∈<br>B→∈<br>is SLR(1) or not? | Understand | This would require the learner to **recall** bottom up parsing technique and **explain** the steps in algorithm for the construction of SLR parsing table and specify is SLR or not. | CO 5 |
| 20 | Consider the grammar bexpr→bexpr or bterm \| btermbterm→bterm and bfactor \| bfactor<br>bfactor→notbfactor \| (bexpr) \|true \| false.<br>Explain whether the grammar is CLR or not? | Understand | This would require the learner to **recall** bottom up parsing technique and **explain** the procedure to check the grammar is CLR or not. | CO 5 |
| **PART - C (PROBLEM SOLVING AND CRITICAL THINKING QUESTIONS)** | | | | |
| 1 | Consider the grammar given below. E → E+T \| T<br>T → T*F \| F F→ (E) \| id.<br>construct LR parsing table for the above grammar .Give the moves of LR parser on id * id + id? | Apply | This would require the learner to **recall** the LR(0) grammar and LR(1) grammar, and **demonstrate** the rules for given grammar and to **construct** the LR parsing table | CO 5 |
| 2 | Identify whether the following grammar is LR(0) with reasons?<br>S → xAy \| xBy \| xAz<br>A → as \| q<br>B → q | Understand | This would require the learner to **recall** the LR(0) grammar and **demonstrate** the rules for given grammar and find out the grammar is LR(0) or | CO 5 |

| | | | | |
|---|---|---|---|---|
| | | | not with specific reasons. | |
| 3 | Construct CLR parsing table for the below grammar?<br>S→ Aa \|<br>aAc \| Bc<br>\|bBa<br>A→d<br>B →d | Apply | This would require the learner to **recall** the CLR parsing method and **demonstrate** the rules for given grammar and to **construct** the CLR parsing table | CO 5 |
| 4 | Identify whether the following grammar is SLR or not with reasons.<br>S→ L = R<br>S→ R<br>L→ * R<br>L→ id<br>R → L. | Understand | This would require the learner to **recall** the LR(0) grammar and **demonstrate** the rules for given grammar and find out the grammar is SLR or not with specific reasons. | CO 5 |
| 5 | Identify whether the following grammar is CLR or not with reasons?<br>S→ AA<br>A →aA \| b | Understand | This would require the learner to **recall** the LR(1) grammar and **demonstrate** the rules for given grammar and find out the grammar is CLR(1) or not with specific reasons. | CO 5 |
| 6 | Construct SLR parsing table for the below grammar?<br>E → E+T \|T<br>T → T*F \| F<br>F→ (E) \|id. | Apply | This would require the learner to **recall** the LR(0) grammar and **demonstrate** the rules for given grammar and to **construct** the SLR parsing table. | CO 5 |
| 7 | The following grammar for if-then-else statements is proposed to remedy the dangling-else ambiguity:<br>Stmt → **if** Expr**then** Stmt<br>\| **if** Expr**then** Stmt**else** Stmt<br>\| **other**<br>Show that how shift and reduce conflicts can be handled in ambiguous grammar. | Understand | This would require the learner to **recall** bottom up parsing technique and **explain** the procedure to handle the ambiguous grammar. | CO 5 |
| 8 | Construct LALR (1) Parsing table for following grammar?<br>S → Aa \|aAc \| Bc \| bBa<br>A →d<br>B →d | Apply | This would require the learner to **recall** the LR(1) grammar and **demonstrate** the rules for given grammar and to **construct** the LALR parsing table | CO 4 |
| 9 | Consider the grammar S→ aSbS \| bSaS\|∈<br>a) Construct the corresponding leftmost derivation and rightmost derivation For abab<br>b) Construct the corresponding parse trees for abab and identify whether the grammar is ambiguous or not. | Apply | This would require the learner to **recall** context free grammars and **explain** the procedure to **construct** derivations. | CO 5 |
| 10 | Construct the FIRST and FOLLOW sets for following grammar?<br>S→ aBDh<br>B → cC<br>C → bC / ∈<br>D → EF<br>E → g / ∈<br>F → f / ∈ | Apply | This would require the learner to **recall** the top down parsing methods and **demonstrate** the rules for given grammar and to **construct** the FIRST and FOLLOW. | CO 4 |

| MODULE-III | | | | |
|---|---|---|---|---|
| SYNTAX-DIRECTED TRANSLATION AND INTERMEDIATE CODE GENERATION | | | | |
| PART - A (SHORT ANSWER QUESTIONS) | | | | |
| 1 | What is the usage of syntax directed definition? | Remember | --- | CO 7 |
| 2 | Define Attribute Grammar? | Remember | --- | CO 7 |
| 3 | List the types of Attribute Grammar? | Remember | --- | CO 7 |
| 4 | Explain syntax directed translation? | Understand | This would require the learner to **recall** Syntax-Directed Translation and **explain** the translations. | CO 7 |
| 5 | Compare synthesized and inherited attributes? | Understand | This would require the learner to **recall** Syntax-Directed Translation and **explain** the synthesized and inherited attributes. | CO 7 |
| 6 | Define L attributed grammar? | Remember | --- | CO 7 |
| 7 | Define S attribute grammar? | Remember | --- | CO 7 |
| 8 | show the Syntax tree for Expression using functions?<br>(a + b) * ( b - c) | Remember | --- | CO 7 |
| 9 | Explain the functions to create nodes of Syntax tree for expression? | Understand | This would require the learner to **recall** Syntax-Directed Translation and **explain** the nodes of Syntax tree. | CO 7 |
| 10 | Define syntax tree? Draw the syntax tree for the assignment statement?<br>a :=b * -c + b * -c. | Remember | --- | CO 7 |
| CIE-II | | | | |
| 11 | Define Translation schemes? | Remember | --- | CO 8 |
| 12 | Define Annotated Parse Tree? | Remember | --- | CO 8 |
| 13 | Explain the three kinds of intermediate representation? | Understand | This would require the learner to **recall** Intermediate Code Generation and **explain** the intermediate forms | CO 8 |
| 14 | What are the benefits of using machine-independent intermediate form? | Remember | --- | CO 8 |
| 15 | What is postfix notation? | Remember | --- | CO 8 |
| 16 | How can you generate three-address code? | Remember | --- | CO 8 |
| 17 | Translate x+y-(a*b)+c into three address code? | Understand | This would require the learner to **recall** Intermediate Code Generation and **convert** to the three address code form. | CO 8 |
| 18 | Discuss back-end and front-end? | Understand | This would require the learner to **recall** phases of a compiler and explain the back-end and front-end. | CO 8 |
| 19 | Define abstract syntax tree? | Remember | --- | CO 8 |
| 20 | List out types of three address code? | Remember | --- | CO 8 |

| PART – B (LONG ANSWER QUESTIONS) | | | | |
|---|---|---|---|---|
| 1 | Explain briefly about syntax directed definition and it types? | Understand | This would require the learner to **recall** Syntax-Directed Translation and **explain** the types of SDD | CO 7 |
| 2 | Explain briefly about Synthesized and Inherited attribute in detail? | Understand | This would require the learner to **recall** Syntax-Directed Translation and **explain** the attributes. | CO 7 |
| 3 | Define translation scheme and write for a<b or b>c? | Remember | --- | CO 7 |
| 4 | Explain briefly about S-attributed and L-attributed grammar in detail? | Understand | This would require the learner to **recall** Syntax-Directed Translation and **explain** the attributes. | CO 7 |
| 5 | Explain how declaration is done in a procedure using syntax directed translation? | Understand | This would require the learner to **recall** Syntax-Directed Translation and **explain** the SDT for procedure. | CO 7 |
| 6 | Explain briefly about postfix Translation Scheme? | Understand | This would require the learner to **recall** Syntax-Directed Translation and **explain** the postfix Translation Scheme | CO 7 |
| 7 | Describe the method of generating syntax directed definition for control Statements? | Remember | --- | CO 7 |
| 8 | Show SDT for the simple assignment statement with example? | Understand | This would require the learner to **recall** Syntax-Directed Translation and **explain** the SDT for simple assignment statement | CO 7 |
| 9 | Explain the construction steps and construct the syntax tree for expression using functions? (m * n + p) + ( m – n + p)? | Understand | This would require the learner to **recall** Syntax-Directed Translation and **explain** the types of three address code. | CO 7 |
| 10 | Explain briefly syntax directed translation into three address code with suitable example? | Understand | This would require the learner to **recall** Syntax-Directed Translation and **explain** the conversion of SDT to three address code. | CO 7 |
| **CIE-II** | | | | |
| 11 | Explain three address codes and mention its types. How would you implement the three address statements? Explain with suitable examples? | Understand | This would require the learner to **recall** Intermediate Code Generation and **explain** the three address code forms. | CO 8 |
| 12 | Explain with an example to generate the intermediate code for the flow of control statements? | Understand | This would require the learner to **recall** Intermediate Code Generation and **explain** the generation of intermediate code for the flow of control statements. | CO 8 |
| 13 | Explain about Quadruple and Triple with its structure? | Understand | This would require the learner to **recall** Intermediate Code Generation and **explain** the quadruple and triple. | CO 8 |
| 14 | Define and represent the Triple, indirect triple and quadruple for the assignment statement?<br>x:= -b + d * -b+d | Remember | --- | CO 8 |

| | | | | |
|---|---|---|---|---|
| 15 | Translate the arithmetic expression a* -(b+c) into<br>a) A syntax tree<br>b) Postfix notation | Understand | This would require the learner to **recall** Intermediate Code Generation and **explain** the arithmetic expression translation into syntax tree & Postfix notation . | CO 8 |
| 16 | Translate the expression – (a + b) * (c + d) + ( a + b +c) into<br>a) quadruples<br>b) triples | Understand | This would require the learner to **recall** Intermediate Code Generation and **explain** the arithmetic expression translation into triple, indirect triple and quadruple. | CO 8 |
| 17 | Show translation scheme for Boolean Expressions with example? | Understand | This would require the learner to **recall** Intermediate Code Generation and **explain** the Boolean Expression translation scheme. | CO 8 |
| 18 | Show translation scheme for Control Flow with example? | Understand | This would require the learner to **recall** Intermediate Code Generation and **explain** the Control Flow. | CO 8 |
| 19 | Translate the expression – (a + b) * (c + d) + ( a + b +c) into<br>a)triples<br>b) indirect triples. | Understand | This would require the learner to **recall** Intermediate Code Generation and **explain** the arithmetic expression translation into triple, indirect triple. | CO 8 |
| 20 | Explain the three address code and draw the abstract tree for the following expressions?<br>(a-b)*c+m-n | Understand | This would require the **recall** Intermediate Code Generation and **explain** the three address code. | CO 8 |
| **PART – C (PROBLEM SOLVING AND CRITICAL THINKING)** | | | | |
| 1 | Construct production rules and semantic actions for S-attributed grammar for the following grammar along with syntax tree and annotated parse tree for the given string a*b-c/d+e?<br>L→E<br>E→ E+T \| E-T \| T<br>T→ T*F \| T/F \|F F→ P-F \| P<br>P→(E)<br>P→ID | Apply | This would require the learner to **recall** Syntax-Directed Translation and **explain** the rules for the conversions and **Construct** the S-attributed grammar. | CO 7 |
| 2 | Construct production rules and semantic actions for the following grammar along with annotated parse tree for the string 9-5+4?<br>expr→ expr + term<br>\| expr – term<br>\| term term→0\|1\|2\|3\|4\|5\|6\|7\|8\|9 | Apply | This would require the learner to **recall** Syntax-Directed Translation and **explain** the rules for the conversions and **Construct** the parse tree. | CO 7 |
| 3 | Construct production rules and semantic actions for the following grammar along with annotated parse tree for the expression: “int a, b, c”?<br>D →T L<br>T →int<br>T → float<br>L→ L1,id L → id | Apply | This would require the learner to **recall** Syntax-Directed Translation and **explain** the rules for the conversions and **Construct** the parse tree. | CO 7 |
| 4 | Construct production rules and semantic actions for the following grammar along | Apply | This would require the learner to **recall** Syntax-Directed | CO 7 |

| | | | | |
|---|---|---|---|---|
| | with annotated parse tree for the string (3+4)*(5+6)?<br>L→E<br>E→ T E→E1+T<br>T→ F T→ T1*F F→ (E)<br>F→ digit | | Translation and **explain** the rules for the conversions and **Construct** the parse tree. | |
| 5 | Construct production rules and semantic actions for the following grammar along with annotated parse tree for the string a-4+c?<br>E→E1+T<br>E→E1-T<br>E→T T→(E)<br>T→id<br>T→ num | Apply | This would require the learner to **recall** Syntax-Directed Translation and **explain** the rules for the conversions and **Construct** the parse treeo-. | CO 7 |
| | **CIE-II** | | | |
| 6 | Construct the three address code and draw the abstract tree for the following expressions?<br>a) (x-y)*z+m-n<br>b) a+(b-c)+(b+c)*(a*e) | Apply | This would require the **recall** Intermediate Code Generation and **explain** the concepts to **construct** the three address code. | CO 8 |
| 7 | Construct the three-address code for the following C program fragment?<br>while(a > b)<br>{<br>if (c < d) x = y +z;<br>else<br>x = y -z;<br>} | Apply | This would require the **recall** Intermediate Code Generation and **explain** the concepts to **construct** the three address code. | CO 8 |
| 8 | Construct triples, Indirect and quadruples of an expression:<br>a = b * - c + b * - c? | Apply | This would require the **recall** Intermediate Code Generation and **explain** the concepts to **construct** the .triples, Indirect and quadruples. | CO 8 |
| 9 | Construct triples, Indirect and quadruples of an expression :<br>x = ( a + b )* - c/d? | Apply | This would require the **recall** Intermediate Code Generation and **explain** the concepts to **construct** triples, Indirect and quadruples. | CO 8 |
| 10 | Why are quadruples preferred over triples in an optimizing compiler with example? | Remember | --- | CO 8 |
| | **MODULE-IV** | | | |
| | **TYPE CHECKING AND RUN TIME ENVIRONMENT** | | | |
| | **PART – A (SHORT ANSWER QUESTIONS)** | | | |
| 1 | List different data structures used for symbol table? | Remember | --- | CO 10 |
| 2 | Define Type checking? | Remember | --- | CO 9 |
| 3 | List the different types of type checking? | Remember | --- | CO 9 |
| 4 | Define Type Expression? | Remember | --- | CO 9 |
| 5 | Explain about the type systems? | Understand | This would require the learner to **recall** type checking and **explain** the type systems | CO 9 |
| 6 | Show the Translation scheme for checking the type of Assignment statement S→id:=E | Remember | --- | CO 9 |

| | | | | |
|---|---|---|---|---|
| 7 | Explain Dynamic type checking? | Understand | This would require the learner to **recall** type checking and **explain** the Dynamic type checking | CO 9 |
| 8 | Define Structural Equivalence? | Remember | --- | CO 9 |
| 9 | What is the Strongly typed language? | Remember | --- | CO 9 |
| 10 | Define Type error? | Remember | --- | CO 9 |
| 11 | Write a short note on static type checking? | Understand | This would require the learner to **recall** type checking and **explain** the static type checking | |
| 12 | Show the Translation scheme for checking the type of Conditional statement - S→if E then S1 | Remember | --- | CO 9 |
| 13 | Show the Translation scheme for checking the type of while statement - S→While E do S1 | Remember | --- | CO 9 |
| 14 | Define Type conversion? | Remember | --- | CO 9 |
| 15 | List the types of type conversion? | Remember | --- | CO 9 |
| 16 | Write about general activation record? | Understand | This would require the learner to **recall** Run Time Environment and **explain** the general activation record. | CO 10 |
| 17 | Define Symbol table? | Remember | --- | CO 10 |
| 18 | Define Dynamic storage allocation? | Remember | --- | CO 10 |
| 19 | Write short note on procedures? | Understand | This would require the learner to **recall** Run Time Environment and **explain** the procedures | CO 10 |
| 20 | Define Activation tree? | Remember | --- | CO 10 |
| 21 | Define stack storage allocation? | Remember | --- | CO 10 |
| 22 | Define static storage allocation? | Remember | --- | CO 10 |
| 23 | Define heap storage allocation? | Remember | --- | CO 10 |
| 24 | Write a short note on parameter passing? | Understand | This would require the learner to **recall** Run Time Environment and **explain** the parameter passing | CO 10 |
| 25 | Define Control stack? | Remember | --- | CO 10 |
| **PART – B (LONG ANSWER QUESTIONS)** | | | | |
| 1 | Explain the specification of a simple type checker | Understand | This would require the learner to **recall** type checking and **explain** the specification of a simple type checker | CO 9 |
| 2 | Define a type expression? Explain the equivalence of type expressions with an appropriate example? | Remember | --- | CO 9 |
| 3 | Explain about reusing the storage space for names? | Understand | This would require the learner to **recall** Run Time Environment and **explain** the procedure for using storage | CO 10 |

| | | | | |
|---|---|---|---|---|
| | | | space efficiently. | |
| 4 | Discuss about all allocation strategies in run-time storage environment? | Understand | This would require the learner to **recall** Run Time Environment and **explain** the allocation strategies. | CO 10 |
| 5 | Explain the data structures used for implementing Symbol Table? | Understand | This would require the learner to **recall** Run Time Environment and **explain** the Symbol Table implementation | CO 10 |
| 6 | Explain Static and Dynamic Checking of types with examples? | Understand | This would require the learner to **recall** type checking and **explain** the Static and Dynamic Checking | CO 9 |
| 7 | Compare the call by value and call by name with examples? | Understand | This would require the learner to **recall** Run Time Environment and **compare** the call by value and by name | CO 10 |
| 8 | Distinguish between static and dynamic storage allocation? | Understand | This would require the learner to **recall** Run Time Environment and **explain** the Static and Dynamic storage allocation | CO 10 |
| 9 | Explain the type checking of expressions? | Understand | This would require the learner to **recall** type checking and **explain** the procedure for expressions | CO 9 |
| 10 | Explain storage organization in runtime environment? | Understand | This would require the learner to **recall** Run Time Environment and **explain** the storage organization | CO 10 |
| 11 | Explain the types of storage allocations? | Understand | This would require the learner to **recall** Run Time Environment and **explain** the storage allocations types | CO 10 |
| 12 | Describe the name and structure equivalence in type expressions? | Understand | This would require the learner to **recall** type checking and **explain** the name and structure equivalence in type expressions | CO 9 |
| 13 | Explain the type checking of control flow statements? | Understand | This would require the learner to **recall** type checking and **explain** the steps for control flow statements | CO 9 |
| 14 | Explain briefly about storage allocation strategies? | Understand | This would require the learner to **recall** Run Time Environment and **explain** the storage allocations strategies | CO 10 |
| 15 | Describe the basic implementation techniques for symbol table? | Understand | This would require the learner to **recall** Run Time Environment and **explain** the symbol table implementation | CO 10 |
| 16 | Explain the calling sequences of activation record? | Understand | This would require the learner to **recall** Run Time Environment and **explain** the calling sequences. | CO 10 |
| 17 | Differentiate ordered, unordered and binary search tree in symbol table? | Understand | This would require the learner to the **recall** Run Time Environment and **explain** binary search tree in symbol | CO 10 |

| | | | | |
|---|---|---|---|---|
| | | | table | |
| 18 | Explain briefly about static storage allocation with block diagram? | Understand | This would require the learner to **recall** Run Time Environment and **explain** the static storage allocation | CO 10 |
| 19 | Differentiate explicit and implicit allocation of memory to variables? | Understand | This would require the learner to **recall** Run Time Environment and **compare** the allocation types of memory to variables | CO 10 |
| 20 | Differentiate stack and heap storage allocation strategies? | Understand | This would require the learner to **recall** Run Time Environment and **explain** the stack and heap storage allocation strategies. | CO 10 |
| **PART – C (PROBLEM SOLVING AND CRITICAL THINKING QUESTIONS)** | | | | |
| 1 | Suppose that the type of each identifier is a sub range of integers, for expressions with operators +, -, *, div and mod, as in Pascal. Explain type- checking rules that assign to each sub expression the sub range its value must lie in? | Understand | This would require the learner to **recall** type checking **explain** how type checking rules implemented for each type of identifier | CO 9 |
| 2 | Explain briefly about Source language issues? | Understand | This would require the learner to **recall** type checking **explain** about Source language issues | CO 9 |
| 3 | Explain briefly about Activation record with block diagram? | Understand | This would require the learner to **recall** run time environment **explain** the Activation record with block diagram | CO 10 |
| 4 | Discuss about variable length data on stack with neat diagram? | Understand | This would require the learner to **recall** run time environment **explain** the variable length data on stack | CO 10 |
| 5 | Explain briefly about heap storage allocation with block diagram? | Understand | This would require the learner to **recall** run time environment **explain** the heap storage allocation | CO 10 |
| 6 | Explain briefly about stack storage allocation with block diagram? | Understand | This would require the learner to **recall** run time environment **explain** the stack storage allocation | CO 10 |
| 7 | Explain briefly about language facilities for dynamic storage allocation? | Understand | This would require the learner to **recall** run time environment **explain** the language facilities for dynamic storage allocation | CO 10 |
| 8 | Describe the parameter passing methods with examples? | Understand | This would require the learner to **recall** run time environment **explain** the various parameter passing methods. | CO 10 |
| 9 | Explain Over loading of Operators & Functions with examples? | Understand | This would require the learner to **recall** run time environment **explain** the Over loading of Operators & Functions | CO 10 |

| | | | | |
|---|---|---|---|---|
| 10 | Differentiate the call by reference and call by copy restore with examples? | Understand | This would require the learner to **recall** run time environment **explain** the call by reference and call by copy restore. | CO 10 |
| **MODULE-V** | | | | |
| **CODE OPTIMIZATION AND CODE GENERATION** | | | | |
| **PART - A (SHORT ANSWER QUESTIONS)** | | | | |
| 1 | List the principle sources of optimization? | Remember | --- | CO 11 |
| 2 | Define the 3 areas of code optimization? | Remember | --- | CO 11 |
| 3 | Explain the techniques used for loop optimization and Reduction in strength? | Understand | This would require the learner to **recall** code optimization and **Explain** the loop optimization methods. | CO 11 |
| 4 | Define constant folding? | Remember | --- | CO 11 |
| 5 | Define Common Sub expressions? | Remember | --- | CO 11 |
| 6 | Explain Dead Code? | Understand | This would require the learner to **recall** code optimization and **Explain** the concept of dead code. | CO 11 |
| 7 | Define local optimization? | Remember | --- | CO 11 |
| 8 | What is Register allocation and assignment? | Remember | --- | CO 12 |
| 9 | Define flow graph and basic block? | Remember | --- | CO 11 |
| 10 | Explain about inner loops? | Understand | This would require the learner to **recall** code optimization and **Explain** the flow graph representation to find the inner loops. | CO 11 |
| 11 | Define a DAG? Mention its Remember? | Remember | --- | CO 12 |
| 12 | Define peephole optimization? | Remember | --- | CO 12 |
| 13 | Define the machine instructions for operations and copy statement? | Remember | --- | CO 11 |
| 14 | Explain global data flow? | Understand | This would require the learner to **recall** code optimization and **Explain** the global data flow. | CO 12 |
| 15 | Explain about live variable analysis? | Understand | This would require the learner to **recall** code optimization and **Explain** the live variable analysis. | CO 11 |
| 16 | Define the term copy propagation? | Remember | --- | CO 11 |
| 17 | Define the term Code motion? | Remember | --- | CO 11 |
| 18 | What is induction variable? | Remember | --- | CO 11 |
| 19 | How do you calculate the cost of an instruction? | Remember | --- | CO 11 |
| 20 | What is the Unreachable Code? | Remember | --- | CO 11 |

| PART - B (LONG ANSWER QUESTIONS) | | | | |
|---|---|---|---|---|
| 1 | Explain the concept of Function-Preserving Transformations? | Understand | This would require the learner to **recall** code optimization and **Explain** the transformations. | CO 11 |
| 2 | Explain Machine dependent code optimization in detail with an example? | Understand | This would require the learner to **recall** code optimization and **Explain** the Machine dependent code optimization. | CO 12 |
| 3 | Write about target code forms and Explain how the instruction forms effect the computation time? | Understand | This would require the learner to **recall** code optimization and **Explain** the how the instruction forms effect the computation time | CO 11 |
| 4 | Explain about machine dependent and machine independent optimization? | Understand | This would require the learner to **recall** code optimization and **Explain** the machine dependent and machine independent optimization | CO 11 |
| 5 | Explain the role of code generator in a compiler? | Understand | This would require the learner to **recall** code generation and **explain** the code generator in a compiler. | CO 12 |
| 6 | Explain in detail the issues in the design of code generator? | Understand | This would require the learner to **recall** code generation and **explain** the issues in the design of code generator | CO 12 |
| 7 | Explain the instructions and address modes of the target machine? | Understand | This would require the learner to **recall** code generation and **explain** the instructions and address modes of the target machine | CO 12 |
| 8 | Explain the principle sources of code optimization in detail? | Understand | This would require the learner to **recall** code optimization and **Explain** the sources of code optimization | CO 11 |
| 9 | Explain the primary structure preserving transformations on basic blocks? | Understand | This would require the learner to **recall** code optimization and **Explain** the structure preserving transformations | CO 11 |
| 10 | Explain peephole optimization in detail? | Understand | This would require the learner to **recall** code optimization and **Explain** the peephole optimization | CO 12 |
| 11 | Define the following<br>i. Copy propagation<br>**ii.** Dead code elimination | Remember | --- | CO 11 |
| 12 | Explain in the DAG representation of the basic block with example? | Understand | This would require the learner to **recall** DAG and **Explain** the representation of the basic blocks | CO 12 |
| 13 | Explain loop optimization in detail with example? | Understand | This would require the learner to **recall** code optimization and **Explain** the loop optimizations | CO 11 |
| 14 | Explain various Global optimization techniques in detail? | Understand | This would require the learner to **recall** code optimization and **Explain** the global optimizations | CO 11 |

| | | | | |
|---|---|---|---|---|
| 15 | Explain Loops in flow graph in detail with example? | Understand | This would require the learner to **recall** code optimization and **Explain** the flow graphs | CO 11 |
| 16 | Explain Local optimization in detail with example? | Understand | This would require the learner to **recall** code optimization and **Explain** the local optimization | CO 11 |
| 17 | Explain Redundant-instructions elimination and Flow-of-control Optimizations? | Understand | This would require the learner to **recall** code optimization and **Explain** optimizations. | CO 11 |
| 18 | Explain the simple code generator with a suitable example? | Understand | This would require the learner to **recall** code generation and **explain** the simple example for code generator. | CO 12 |
| 19 | Explain the procedure to detect induction variable and dead code elimination with example? | Understand | This would require the learner to **recall** code optimization and **Explain** optimizations. | CO 11 |
| 20 | Explain briefly about register allocation and assignment? | Understand | This would require the learner to **recall** code generation and **explain** the register allocation. | CO 12 |
| **PART – C (PROBLEM SOLVING AND CRITICAL THINKING)** | | | | |
| 1 | Construct the code sequence generated by the simple code generation algorithm for x*y+(m-k)-(g+b) | Apply | This would require the learner to **recall** code generation and **explain** the procedure to **Construct** the code sequence. | CO 12 |
| 2 | Construct target code for the given program segments:<br>main()<br>{<br>int i=4,j; j = i + 5;<br>} | Apply | This would require the learner to **recall** code generation and **explain** the procedure to **Construct** the target code. | CO 12 |
| 3 | Consider the following basic block of 3-address instructions .Construct target code for the source language statement and finds its cost.<br>a := b + c x := a + b b := a – d c := b + c d := a –d y := a – d | Apply | This would require the learner to **recall** code generation and **explain** the procedure to **Construct** the target code. | CO 12 |
| 4 | Identify the register descriptor target code for the source language Statement and its cost.<br>(a-b) + (a-c) + (a-c) | Apply | This would require the learner to **recall** code generation and **explain** the procedure to **Construct** the target code. | CO 12 |
| 5 | Consider the following part of code.<br>int main()<br>{<br>intn,k=0; scanf("%d",&n);<br>for(i=2;i<n;i++)<br>{<br>if(n%I),==0)break;<br>} k=1;<br>if(i==n)<br>printf("number is prime");<br>else<br>printf("number is not printed");<br>}<br>Identify the basic block in the given | Apply | This would require the learner to **recall** code generation and **explain** the procedure to **Construct** the basic blocks. | CO 12 |

| | | | | |
|---|---|---|---|---|
| | program | | | |
| 6 | Construct the DAG for the following basic block.<br>D:=B*C<br>E:=A+B<br>B:=B+C<br>A:=E-D | Apply | This would require the learner to **recall** code generation and **explain** the procedure to **Construct** the DAG. | CO 12 |
| 7 | Construct basic block for following code<br>void quicksort(m, n)<br>int m, n;<br>{<br>int i, j;<br>if (n <= m )<br>return; /* fragment begins here */<br>i = m-1;<br>j = n;<br>v = a[n]; while(1)<br>{<br>do<br>i = i+1;<br>while( a[i] < v);<br>do<br>j = j-1;<br>while( a[j] > v ); if( i >= j )<br>break;<br>x = a[i];<br>a[i] = a[j]; a[j] =x;<br>}<br>x = a[i]; a[i] = a[n];<br>a[n]= x; /* fragment ends here */<br>quicksort(m, j);<br>quicksort(i+1, n);<br>}. | Apply | This would require the learner to **recall** code generation and **explain** the procedure to **Construct** the basic blocks. | CO 12 |
| 8 | Construct DAG and explain the procedure for the conversion. a+b*(a+b)+c+d | Apply | This would require the learner to **recall** code generation and **explain** the procedure to **Construct** the DAG. | CO 12 |
| 9 | Explain role of DAG representation in optimization with example? | Understand | This would require the learner to **recall** code generation and **explain** the DAG and its role in optimization. | CO 12 |
| 10 | Construct the basic block and flow graph for the following code<br>begin<br>prod :=0; i:=1;<br>do begin<br>prod :=prod+ a[i] * b[i];<br>i :=i+1;<br>end<br>while i <= 20 end | Apply | This would require the learner to **recall** code generation and **explain** the procedure to **Construct** the basic blocks. | CO 12 |
| 11 | Construct optimal machine code for the following C program. main()<br>{<br>inti,a[10]; while(i<=10)<br>a[i]=0;<br>} | Apply | This would require the learner to **recall** code generation and **explain** the procedure to **Construct** the optimal machine code. | CO 12 |