INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

COMPUTER SCIENCE AND ENGINEERING

TUTORIAL QUESTION BANK

Course Name	**COMPILER DESIGN**
Course Code	A50514
Class	III B.Tech I Semester
Branch	Computer Science and Engineering
Year	2017– 2018
Course Coordinator	Ms. B Ramyasree, Assistant Professor.
Course Faculty	Ms. N Mamatha, Assistant Professor, Mr. N Poornachandra Rao, Assistant Professor.

OBJECTIVES

To meet the challenge of ensuring excellence in engineering education, the issue of quality needs to be addressed, debated and taken forward in a systematic manner. Accreditation is the principal means of quality assurance in higher education. The major emphasis of accreditation process is to measure the outcomes of the program that is being accredited.

In line with this, Faculty of Institute of Aeronautical Engineering, Hyderabad has taken a lead in incorporating philosophy of outcome based education in the process of problem solving and career development. So, all students of the institute should understand the depth and approach of course to be taught through this question bank, which will enhance learners learning process.

S. No.	Questions	Bloom's Taxonomy Level	Course Outcome
UNIT-I			
PART – A (SHORT ANSWER QUESTIONS)			
1	**Define** Complier briefly?	Understand	1
2	**Explain** the cousins of compiler?	Understand	1
3	**Define** the two main parts of compilation? What they perform?	Understand	1
4	**Explain how** many phases does analysis consists?	Understand	1
5	**Define** and explain the Loader?	Remember	3
6	**Explain** about preprocessor?	Remember	1
7	**State** the general phases of a compiler?	Understand	3
8	**State** the rules and define regular expression?	Remember	2
9	**Explain** a lexeme and define regular sets?	Remember	2
10	**Explain** the issues of lexical analyzer?	Understand	2
11	**State** some compiler construction tools?	Understand	3

S. No.	Questions	Bloom's Taxonomy Level	Course Outcome
12	**Define** the term Symbol table?	Understand	1
13	**Define** the term Interpreter?	Remember	1
14	**Define** the term Tokens in lexical analysis phase?	Understand	1
15	**Explain** about error Handler?	Understand	1
16	**Define** a translator and types of translator?	Understand	1
17	**Explain** about parser and its types?	Understand	1
18	**Construct** NFA for (a/b)* and convert into DFA?	Remember	2
19	**Define** bootstrap and cross compiler?	Understand	1
20	**Define** pass and phase?	Understand	3
21	**Analyze** the output of syntax analysis phase? what are the three general types of parsers for grammars?	Remember	1
22	**List** the different strategies that a parser can employ to recover from a syntactic error?	Understand	1
23	**Explain** the goals of error handler in a parser?	Understand	3
24	**Explain** why will you define a context free grammar?	Remember	3
25	**Define** context free language. When will you say that two CFGs are equal?	Remember	2
26	**Give** the definition for leftmost and canonical derivations?	Understand	4
27	**Define** a parse tree?	Understand	1
28	**Explain** an ambiguous grammar with an example?	Apply	1
29	**When** will you call a grammar as the left recursive one?	Apply	4
30	**List** different types of compiler?	Remember	1
PART – B (LONG ANSWER QUESTIONS)			
1	**Define** compiler? State various phases of a compiler and explain them in detail.	Understand	1
2	**Explain** the various phases of a compiler in detail. Also writedown the output for the following expression after each phase a: =b*c-d.	Apply	1
3	**Explain** the cousins of a Compiler? Explain them in detail.	Understand	1
4	**Describe** how various phases could be combined as a pass in a compiler? Also briefly explain Compiler construction tools.	Remember	3
5	**For** the following expression Position:=initial+ rate*60 Write down the output after each phase	Apply	1
6	**Explain** the role Lexical Analyzer and issues of Lexical Analyzer.	Remember	1
7	**Differentiate** the pass and phase in compiler construction?	Remember	1
8	**Explain** single pass and multi pass compiler with example?	Understand	1
9	**Define** bootstrapping concept in brief?	Understand	1
10	**Explain** the general format of a LEX program with example?	Understand	3
11	**Construct** the predictive parser the following grammar: S->(L)|a L->L,S|S Construct the behavior of the parser on the sentence (a, a) using the grammar specified above	Apply	4

<table>
<tr><th>S. No.</th><th>Questions</th><th>Bloom's Taxonomy Level</th><th>Course Outcome</th></tr>
<tr><td>12</td><td><b>Explain</b> the algorithm for finding the FIRST and FOLLOW positions for a given non-terminal.
Consider the grammar,
E ->TE
E->+TE|@
T ->FT
T->*FT|@
F->(E)|id.
<b>Construct</b> a predictive parsing table for the grammar given above. Verify whether the input string id + id * id is accepted by the grammar or not.</td><td>Understand</td><td>4</td></tr>
<tr><td>13</td><td><b>Prepare</b> the predictive parser for the following grammar:
S->a|b|(T)
T ->T, S|S
Write down the necessary algorithms and define FIRST and FOLLOW. Show the behavior of the parser in the sentences.
i.(a,(a,a))
ii.(((a,a),a,(a),a)</td><td>Apply</td><td>4</td></tr>
<tr><td>14</td><td><b>Explain</b> operator grammar? Draw the precedence function graph for the following table.
<table>
<tr><th></th><th>A</th><th>(</th><th>)</th><th>,</th><th>$</th></tr>
<tr><td>a</td><td></td><td></td><td>></td><td>></td><td>></td></tr>
<tr><td>(</td><td><</td><td><</td><td>=</td><td><</td><td></td></tr>
<tr><td>)</td><td></td><td></td><td>></td><td>></td><td>></td></tr>
<tr><td>,</td><td><</td><td><</td><td>></td><td>></td><td></td></tr>
<tr><td>$</td><td><</td><td><</td><td></td><td></td><td></td></tr>
</table></td><td>Understand</td><td>4</td></tr>
<tr><td>15</td><td><b>Analyze</b> whether the following grammar is LR(1) or not. Explain your answer with reasons.
<b>S-> L,R</b>
<b>S-> R</b>
<b>L -> * R</b>
<b>L-> id</b>
<b>R-> L.</b></td><td>Analysis</td><td>4</td></tr>
<tr><td>16</td><td><b>Difference</b> between nondeterministic and deterministic finite automata</td><td>Understand</td><td>4</td></tr>
<tr><td>17</td><td><b>Construct</b> regular grammar from regular expression</td><td>Understand</td><td>4</td></tr>
<tr><td>18</td><td><b>Explain</b> the problems in top down parsing</td><td>Understand</td><td>4</td></tr>
<tr><td>19</td><td><b>Explain</b> top down parsing algorithm in detail</td><td>Understand</td><td>4</td></tr>
<tr><td>20</td><td><b>Demonstrate</b> left factoring with example</td><td>Understand</td><td>4</td></tr>
<tr><td colspan="4"><b>PART – C (PROBLEM SOLVING AND CRITICAL THINKING QUESTIONS)</b></td></tr>
<tr><td>1</td><td>Consider the following fragment of C code:
float i, j;
i = i*70+j+2;
<b>Write</b> the output at all phases of the compiler for above C code.</td><td>Apply</td><td>1</td></tr>
</table>

S. No.	Questions	Bloom's Taxonomy Level	Course Outcome
2	**Construct** an NFA for regular expression R= (aa | b) * ab convert it into an equivalent DFA.	Remember	2
3	**Describe** the languages denoted by the following regular expressions. i. (0+1)*0(0+1)(0+1) ii. 0*10*10*10*	Remember	2
4	**Explain** with one example how LEX program perform lexical analysis for the following PASCAL patterns Identifiers, Comments, Numerical constants, Keywords, Arithmetic operators?	Apply	3
5	**Check** whether the following grammar is a LL(1)grammar S-> iEtS|iEtSeS|a E-> b Also define the FIRST and FOLLOW.	Apply	4
6	**Consider** the grammar below E->E+E|E-E|E*E|E/E|a|b Obtain left most and right most derivation for the string a+b*a+b.	Apply	4
7	**Define** ambiguous grammar? Test whether the following grammar is ambiguous or not. E->E+E|E-E|E*E|E/E|E↑|(E)|-E|id	Apply	4
8	**State** the limitations of recursive descent parser?		
9	**Convert** the following grammar into LL(1)grammar S->ABC A->aA|C B->b C->c.	Apply	4
10	**Write** a recursive descent parser for the grammar. bexpr->bexpr or bterm|bterm bterm->bterm and bfactor|bfactor bfactor->notebfactor|(bexpr)|true|false. Where ,or, and , not,(,),true, false are terminals of the grammar.	Apply	4
UNIT – II			
PART – A (SHORT ANSWER QUESTIONS)			
1	**Define** the term handle used in operator precedence?	Understand	5
2	**Define** LR(0) items in bottom up parsing?	Remember	5
3	**State** the disadvantages of operator precedence parsing?	Remember	5
4	**Explain** LR(k) parsing stands for ?	Understand	5
5	**Explain** why LR parsing is attractive one and explain?	Understand	5
6	**Define** goto function in LR parser with an example?	Understand	5
7	**Explain** why SLR and LALR are more economical to construct Canonical LR?	Understand	5
8	**Explain** about handle pruning?	Understand	5
9	**Explain** types of LR parsers?	Understand	5
10	**List** down the conflicts during shift-reduce parsing.	Remember	5
11	**Define** shift reduce parsing in detail	Understand	5
12	**Explain** conflicts in shift reduce parsing	Understand	5
13	**Explain** reduce conflicts with example	Understand	5
14	**Explain** precedence relations in detail	Understand	5

S. No.	Questions	Bloom's Taxonomy Level	Course Outcome
15	**Define** operator grammar with example	Understand	5
16	**Consider** the grammar E -> E + E|E *E|(E)| id 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.	Apply	5
17	i) **State** shift-reduce parsing? Explain in detail the conflicts that may occur during shift-reduce parsing. ii)For the grammar given below, calculate the operator precedence relation and the precedence functions E-> E + E|E- E|E * E|E / E|E E|(E)|-E|id	Understand	5
18	**Prepare** a canonical parsing table for the grammar given below S-> CC C->cC|d	Analysis	5
19	**Analyze** whether the following grammar is SLR(1) or not. Explain your answer with reasons. S -> L,R S -> R L -> * R L -> id R -> L.	Apply	5
20	i) **Consider** the grammar given below. E -> E + T E -> T T -> T * F T -> F F -> (E) F -> id Prepare LR parsing table for the above grammar .Give the moves of LR parser on id * id + id ii) Briefly explain error recovery in LR parsing.	Apply	5
21	**Explain** handle pruning in detail with example	Understand	4
22	**Demonstrate** stack implementation in implementation of shift reduce Parsing	Understand	4
23	**Explain** ways to determine precedence relations between pair of terminals	Understand	4
24	**Explain** operator precedence parsing algorithm	Understand	4
25	**Explain** LR parsers in detail with example	Understand	4
PART – B (LONG ANSWER QUESTIONS)			
1	Consider the grammar E->E + E|E *E|(E)| id.**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.	Apply	5
2	i) **State** shift-reduce parsing? Explain in detail the conflicts that may occur during shift-reduce parsing. ii)For the grammar given below, **calculate** the operator precedence relation and the precedence functions E -> E + E|E- E|E * E|E / E|E E|(E)|-E|id	Understand	5
3	**Prepare** a canonical parsing table for the grammar given below S-> CC C -> cC|d	Analysis	5

<table>
<tr><th>S. No.</th><th>Questions</th><th>Bloom's Taxonomy Level</th><th>Course Outcome</th></tr>
<tr><td>4</td><td><b>Analyze</b> whether the following grammar is SLR(1) or not. Explain your answer with reasons.
S->L,R
S-> R
L-> * R
L-> id
R -> L.</td><td>Apply</td><td>5</td></tr>
<tr><td>5</td><td><b>Consider</b> the grammar given below.
E -> E + T
E -> T
T -> T * F
T -> F
F->(E)
F-> id
Prepare LR parsing table for the above grammar .Give the moves of LR parser on id * id + id
ii) <b>Briefly</b> explain error recovery in LR parsing.</td><td>Apply</td><td>5</td></tr>
<tr><td>6</td><td><b>Explain</b> handle pruning in detail with example</td><td>Understand</td><td>4</td></tr>
<tr><td>7</td><td><b>Demonstrate</b> stack implementation in implementation of shift reduce Parsing</td><td>Understand</td><td>4</td></tr>
<tr><td>8</td><td><b>Explain</b> ways to determine precedence relations between pair of terminals</td><td>Understand</td><td>4</td></tr>
<tr><td>9</td><td><b>Explain</b> operator precedence parsing algorithm</td><td>Understand</td><td>4</td></tr>
<tr><td>10</td><td><b>Explain</b> LR parsers in detail with example</td><td>Understand</td><td>4</td></tr>
<tr><td colspan="4"><b>PART – C (PROBLEM SOLVING AND CRITICAL THINKING QUESTIONS)</b></td></tr>
<tr><td>1</td><td><b>Explain</b> the common conflicts that can be encountered in a shift-reduce parser?</td><td>Apply</td><td>5</td></tr>
<tr><td>2</td><td><b>Explain</b> briefly, precedence functions. Construct the precedence graph using the following precedence tables.
<table>
<tr><th></th><th>+</th><th>*</th><th>)</th><th>Id</th><th>$</th></tr>
<tr><td>f</td><td>2</td><td>3</td><td>4</td><td>4</td><td>0</td></tr>
<tr><td>g</td><td>1</td><td>3</td><td>4</td><td>5</td><td>0</td></tr>
</table></td><td>Apply</td><td>5</td></tr>
<tr><td>3</td><td><b>Explain</b> LALR parsing, justify how it is efficient over SLR parsing.</td><td>Remember</td><td>5</td></tr>
<tr><td>4</td><td>Analyze whether the following grammar is CLR(1) or not. Explain your answer with reasons
S -> L,R
S->R
L-> * R
L-> id
R -> L.</td><td>Analysis</td><td>5</td></tr>
<tr><td>5</td><td><b>Discuss</b> error recovery in LL and LR parsing.</td><td>Remember</td><td>5</td></tr>
</table>

S. No.	Questions	Bloom's Taxonomy Level	Course Outcome
6	**Construct** SLR (1) Parsing table for following grammar s-> xAy/xBy/xAz A->as/q B->q	Remember	2
7	**Construct** SLR (1) Parsing table for following grammar s->0s0/1s1/10	Remember	2
8	**Construct** SLR (1) Parsing table for following grammar s->aSbS/bsas/E	Remember	2
9	**Construct** LALR (1) Parsing table for following grammar s->Aa/bAc/dc/bda A->d	Remember	2
10	**Construct** LALR (1) Parsing table for following grammar s->Aa/aAc/Bc/bBa A->d B->d	Remember	2
UNIT – III			
PART – A (SHORT ANSWER QUESTIONS)			
1	**State** the benefits of using machine-independent intermediate form?	Remember	8
2	**List** the three kinds of intermediate representation?	Understand	8
3	**Explain how** can you generate three-address code?	Understand	8
4	**Define** syntax tree? Draw the syntax tree for the assignment statement. a :=b * -c + b * -c.	Apply	6
5	**Explain** postfix notation?	Remember	8
6	**Explain** the usage of syntax directed definition?	Apply	7
7	**Define** abstract or syntax tree?	Understand	7
8	**Show** the DAG for a: =b *-c + b * -c?	Apply	7
9	**Translate** a or b and not c into three address code?	Apply	8
10	**Define** basic blocks?	Understand	9
11	**Discuss** back-end and front-end?	Understand	8
12	**Define** the primary structure preserving transformations on basic blocks?	Understand	8
13	**List** common methods for associating actual and formal parameters?	Understand	8
14	**List** various forms of target programs?	Remember	8
15	**Define** back patching?	Understand	8
16	**List** different data structures used for symbol table?	Remember	9
17	**Explain** the steps to search an entry in the hash table?	Understand	9
18	**List** the different types of type checking? Explain?	Understand	7
19	**Explain** general activation record?	Understand	9
20	**State** the difference between heap storage and hash table?	Understand	9
PART – B (LONG ANSWER QUESTIONS)			
1	**Explain** with an example to generate the intermediate code for the flow of control statements?	Apply	8
2	**List** the various ways of calling the procedures? Explain in detail?	Analysis	6

S. No.	Questions	Bloom's Taxonomy Level	Course Outcome
3	**Explain** 3addresscodes and mention its types. How would you implement the three address statements? Explain with suitable examples?	Apply	8
4	**Explain** how declaration is done in a procedure using syntax directed translation?	Apply	7
5	a) **Write** a note on the specification of a simple type checker. b) **Define** a type expression? Explain the equivalence of type expressions with an appropriate example.	Analysis	7

6	**Generate** the three-address code for the following C program fragment
	while(a > b)
	{
	if (c < d)
	x = y + z;
	else
	x = y - z;
	}
		Understand	8

7	**Generate** the code for the following C statements using its equivalent three address code.
	a = b + 1
	x = y+3
	y = a/b
	a = b+c
		Understand	8

8	**Describe** the method of generating syntax directed definition for control Statements?	Understand	7
9	**Explain** procedure calls with suitable example?	Understand	7
10	**Explain** Intermediate code generation for Basic block, Control Flow and Boolean Expressions?	Apply	8
11	**Write** about Quadruple and Triple with its structure?	Apply	8
12	**Explain** different schemes of storing name attribute in symbol table.	Understand	9
13	**Write** the advantages and disadvantages of heap storage allocation strategies?	Apply	9
14	**Distinguish** between static and dynamic storage allocation?	Understand	4
15	**Differentiate** between stack and heap storage?	Understand	4
16	**Demonstrate** semantic actions in semantic analysis	Understand	4
17	**Explain** translations on parse tree semantic analysis	Understand	4
18	**Explain** type checking in semantic analysis	Understand	4
19	**Explain** symbol table management in compiler design	Understand	4
20	**Demonstrate** hash tables by symbol table management	Understand	4
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. Write type-checking rules that assign to each sub expression the sub range its value must lie in.	Understand	7

S. No.	Questions	Bloom's Taxonomy Level	Course Outcome
2	**Define** type expression? Write type expression for the following type i.Functions whose domains are functions from integers to pointers to integers and whose ranges are records consisting of an integer and a character.	Understand	7
3	**Write** an S-attributed grammar to connect the following with prefix rotator. L→E E→ E+T|E-T|T T→ T*F|T/F|F F→ P↑F|P P→ (E) P→ ID	Apply	7
4	**Construct** triples of an expression: a *- (b + c).	Apply	8
5	**Explain** SDD for Boolean expression with and without back patching?	Remember	7
6	**Explain why** are quadruples preferred over triples in an optimizing compiler?	Remember	8
7	**Explain** about reusing the storage space for names?	Remember	9
8	**Define** self-organizing lists? How can this be used to organize a symbol table? Explain with an example?	Apply	9
9	**Discuss** and analyze about all allocation strategies in run-time storage environment?	Understand	9
10	**Define** activation records? Explain how it is related with run-time storage organization?	Remember	9
11	Only one occurrence of each object is allowable at a given moment during program execution. **Justify** your answer with respect to static allocation?	Apply	9
12	**Explain** the use of Symbol table in compilation process? List out various attributes stored in the symbol table?	Understand	9
13	**List** the advantages and disadvantages of Static storage allocation strategies?	Understand	9
14	**Explain** the data structure used for implementing Symbol Table?	Understand	9
15	**Explain** the following: i) Static and Dynamic Checking of types ii) Over loading of Operators & Functions	Understand	7
UNIT – IV			
PART – A (SHORT ANSWER QUESTIONS)			
1	**Explain** the principle sources of optimization?	Understand	10
2	**Explain** the patterns used for code optimization?	Understand	10
3	**Define** the 3 areas of code optimization?	Understand	10
4	**Define** local optimization?	Understand	10
5	**Define** constant folding?	Understand	10
6	**List** the advantages of the organization of code optimizer?	Understand	10
7	**Define** Common Sub expressions?	Understand	10
8	**Explain** Dead Code?	Understand	10

S. No.	Questions	Bloom's Taxonomy Level	Course Outcome
9	**Explain** the techniques used for loop optimization and Reduction in strength?	Understand	12
10	**Mention** the issues to be considered while applying the techniques for code Optimization?	Understand	12
11	**List** the different data flow properties?	Understand	11
12	**Explain** inner loops?	Understand	11
13	**Define** flow graph?	Understand	11
14	**Define** a DAG? Mention its Apply?	Understand	12
15	**Define** peephole optimization?	Understand	12
16	**Explain** machine instruction for operations and copy statement?	Understand	12
17	**Analyze** global data flow?	Understand	11
18	**Explain** about live variable analysis**?**	Understand	10
19	**Define** the term copy propagation?	Understand	11
20	**Explain** data flow equation?	Understand	11
PART – B (LONG ANSWER QUESTIONS)			
1	**Explain** the principle sources of code optimization in detail?	Understand	10
2	**Explain** peephole optimization?	Understand	10
3	**Discuss** about the following i. Copy propagation ii. Dead code elimination iii. Code motion	Understand	10
4	**Explain** in the DAG representation of the basic block with example.	Understand	11
5	**Explain** Local optimization and loop optimization in detail	Understand	11
6	**Write** about Data Flow Analysis of structural programs?	Understand	12
7	**Explain** various Global optimization techniques in detail?	Understand	12

8	**Generate** target code for the given program segments:
	main()
	{
	int i=4,j;
	j = i + 5;
	}
		Apply	11

9	**Discuss** algebraic simplification and reduction in strength?	Understand	11
10	**Explain** the various source language issues?	Understand	10
11	**Explain** in detail the issues in design of a code generator?	Understand	13
12	**Demonstrate** the simple code generator with a suitable example?	Apply	13
13	**List** the different storage allocation strategies? Explain.	Understand	12
14	(a) **Write** the procedure to detect induction variable with example? (b) With example **Explain** dead code elimination?	Understand	11
15	(a) **Explain** how loop invariant computation can be eliminated? (b) **Explain** how "Redundant sub-expression eliminates" can be done in a given program?	Understand	11
16	**Explain** reachable code in code optimization	Understand	11
17	**Explain** characteristics of peep hole optimization	Understand	11

S. No.	Questions	Bloom's Taxonomy Level	Course Outcome
18	**Explain** depth first search in data flow analysis	Understand	11
19	**Explain** node splitting in data flow analysis	Understand	11
20	**Explain** depth first ordering in iterative algorithms	Understand	11
PART – C (PROBLEM SOLVING AND CRITICAL THINKING QUESTIONS)			
1	**Explain** how loop invariant computation can be eliminated?	Apply	10
2	**Describe** the procedure to compute in and out values using data flow equations for reaching definition in structured programs?	Apply	11

3	**Consider** the following part of code.
	int main()
	{
	int n,k=0;
	scanf("%d",&n);
	for(i=2;i<n;i++)
	{
	if((n%I)==0)break;
	}
	k=1;
	if(i==n)
	printf("number is prime");
	else
	printf("number is not printed");
	}
	Identify the basic blocks in the given program & Draw the domination tree for the program
		Understand	12

4	**Construct** the DAG for the following basic block.
	D:=B*C
	E:=A+B
	B:=B+C
	A:=E-D
		Apply	11

5	**Consider** the following program which counts the prime from 2 to n using the sieve method on a suitable large array, begin read n
	for i:=2 to n do
	a[i]:=true
	count=0;
	for i:=2 to n**.5 do
	if a[i]then
	begin
	count:=2*I to n j=j+1 do
	a[j]:=false end
	i. print count end
	ii. Propagate out copy statements wherever possible.
	iii. Is loop jamming possible? If so, do it.
	iv. Eliminate the induction variables wherever possible
		Apply	12

6	**Write** an algorithm to eliminate induction variable?	Apply	10
7	**Explain** how the following expression can be converting in a DAG. a+b*(a+b)+c+d	Apply	11

S. No.	Questions	Bloom's Taxonomy Level	Course Outcome
8	**State** loop invariant computations? Explain how they affect the efficiency of a program?	Understand	10
9	**Explain** how "Redundant sub-expression Eliminates" can be done at global level in a given program?	Understand	10
10	**Explain** role of DAG in optimization with example?	Understand	11
	UNIT – V		
	PART – A (SHORT ANSWER QUESTIONS)		
1	**Explain** about machine dependent and machine independent optimization?	Remember	14
2	**Explain** the role of code generator in a compiler?	Understand	13
3	**Write** in detail the issues in the design of code generator.	Apply	13

4	**Show** the code sequence generated by the simple code generation Algorithm
	u := a – c
	v := t + u
	d := v + u//d
		Apply	13

5	**Explain** the instructions and address modes of the target machine?	Understand	14
6	**Identify** the register descriptor target code for the source language statement "(a-b) + (a-c) + (a-c);" The 3AC for this can be written as t := a – b	Understand	13
7	**Mention** the properties that a code generator should possess.	Apply	13
8	**Explain** how do you calculate the cost of an instruction?	Understand	14
9	**Explain** how will you map names to values?	Understand	14
10	**Generate** the code for x: =x+1 for target machine?	Understand	14
11	**Explain** the input taken by code generation algorithm	Understand	13
12	**Mention** the applications of DAG	Apply	13
13	**Describe** register descriptors in detail	Understand	14
14	**Describe** address descriptors in detail	Understand	14
15	**Demonstrate** global register allocation with example	Understand	14
	PART – B (LONG ANSWER QUESTIONS)		

1	a) **Explain** the concept of object code forms?
	b) **Generate** optimal machine code for the following C program.
	main()
	{
	int i, a[10];
	while (i<=10)
	a[i] =0;
	}
		Apply	13

2	**Explain** Machine dependent code optimization in detail with an example?	Understand	14

S. No.	Questions	Bloom's Taxonomy Level	Course Outcome
3	(a) **Discuss** various object code forms? (b) **Write** a short note on code generating algorithms?	Understand	13
4	**Write** about target code forms and explain how the instruction forms effect the computation time?	Understand	14

5	**Consider** the following basic block of 3-address instructions:
	a := b + c
	x := a + b
	b := a – d
	c := b + c
	d := a – d
	y := a – d
	Write the next-use information for each line of the basic block?
		Apply	13

6	**Demonstrate** register allocation by graph coloring	Understand	14
7	**Explain** the steps involved in Dag construction	Understand	14
8	**Demonstrate** code generation algorithm in detail	Understand	14
9	**Explain** the principle of dynamic programming in detail	Understand	14
10	**Explain** code generation by tree rewriting in detail	Understand	14
PART – C (PROBLEM SOLVING AND CRITICAL THINKING QUESTIONS)			
1	**Explain** how the instruction forms effect the computation time?	Apply	13
2	**Explain** how the nature of the object code is highly dependent on the machine and the operating system?	Apply	13
3	**Explain** why Next-use information is required for generating object code?	Apply	14
4	Efficient code generation requires the Remember of internal architecture of the target machine. **Justify** your answer with an Example?	Understand	13

5	**Generate** optimal machine code for the following wing c program.
	main()
	{
	int i,a[10];
	while(i<=10)
	a[i]=0;
	}
		Apply	14

6	**Generate** 3 address code for below code X = (a+b)- /((c+d)-e)	Apply	13
7	**Generate** 3 address code for below code For(i=1;i<=10;i++) If(a<b) then x = y + z	Apply	13

8	**Generate** 3 address code for below code
	If a < b then
	While c > d do
	x = x+y
	else
	do
	p = p+q
	while e<=f
		Apply	13

S. No.	Questions	Bloom's Taxonomy Level	Course Outcome

9	**Generate** 3 address code for below code
	X = 1
	X = y
	X = x++
		Apply	13

10	**Generate** 3 address code for below code
	main( )
	{
	int i;
	int a[10];
	While(i<=0)
	a[i]=0;
	}
		Apply	13

Prepared by: Ms B Ramyasree, Assistant Professor,
Ms N Mamatha, Assistant Professor,
Mr N Poornachandra Rao, Assistant Professor.

HOD, COMPUTER SCIENCE AND ENGINEERING