NITTE EDUCATION TRUST | NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY BELAGAVI,
Accredited by NAAC-"A" Grade, approved by AICTE, New Delhi.
Yelahanka,Bangalore-64

Department of Computer Science and Engineering

(NBA Accredited till 30/06/2020)

Subject: Compiler Design (14CS73) **Semester: VII Sem BE** **Credits: 4**

Question Bank 2019-20

<table>
<tr><th colspan="5">UNIT-I</th></tr>
<tr><th>Sl No</th><th>Questions</th><th>Marks</th><th>CO/PO</th><th>Bloom's level</th></tr>
<tr><td>1.</td><td>Compare language processor, compiler, interpreter and hybrid compiler with diagrams to show their working.</td><td>06</td><td>CO1/3,5</td><td>L4</td></tr>
<tr><td>2.</td><td>With a neat diagram explain the structure of a compiler.</td><td>10</td><td>CO1/3,5</td><td>L2</td></tr>
<tr><td>3.</td><td>Differentiate between compilers and interpreters.</td><td>04</td><td>CO1/3,5</td><td>L4</td></tr>
<tr><td>4.</td><td>Describe the science of building a compiler and identify the design objectives for compiler optimization.</td><td>08</td><td>CO1/3,5</td><td>L2</td></tr>
<tr><td>5.</td><td>Provide functions of Lexical Analysis. List and explain the job of lexical analyzer.</td><td>05</td><td>CO1/3,5</td><td>L1</td></tr>
<tr><td>6.</td><td>Provide overview of LEX tool. Explain Lex conventions with an example.</td><td>06</td><td>CO1/3,5</td><td>L2</td></tr>
<tr><td>7.</td><td>Identify the features of Syntax Analysis. Illustrate with an example.</td><td>05</td><td>CO1/3,5</td><td>L3</td></tr>
<tr><td>8.</td><td>Describe intermediate code generation. Explain with an example.</td><td>05</td><td>CO1/3,5</td><td>L2</td></tr>
<tr><td>9.</td><td>List and explain the applications of compiler technology.</td><td>09</td><td>CO1/3,5</td><td>L2</td></tr>
<tr><td>10.</td><td>Briefly explain the role of Lexical Analyzer with a neat diagram.</td><td>08</td><td>CO1/3,5</td><td>L2</td></tr>
<tr><td>11.</td><td>For the given token table below, construct transition diagrams.
<table><tr><td>Token</td><td>Code</td><td>Value</td></tr><tr><td>Begin</td><td>1</td><td></td></tr><tr><td>End</td><td>2</td><td></td></tr></table></td><td>10</td><td>CO1/3,5</td><td>L6</td></tr>
</table>

If	3	
Then	4	
Else	5	
Identifier	6	Pointer to symbol table
<	RELOP	LT
<=	RELOP	LE
=	RELOP	EQ
<>	RELOP	NE
>	RELOP	GT
>=	RELOP	GE

12.	Explain input buffering and justify its significance.	05	CO1/3,5	L5
13.	Describe the types of errors in compilation process.	03	CO1/3,5	L2
14.	Construct transition diagram to recognize i) relational operators <=, <>, >=, = ii) Identifiers and keywords iii) white spaces iv) unsigned numbers Also show the function executed at each final state (accept state).	10	CO1/3,5	L6
15.	Discuss various kinds of translators.	06	CO1/3,5	L1
16.	Provide an overview of compiler construction tools.	06	CO1/3,5	L2
17.	Describe in brief all the phases of a compiler. With a neat diagram show the output of each phase for the expression, Position = Initial + Rate*20	10	CO1/3,5	L3
18.	Explain LEX tool and justify its uses.	07	CO1/3,5	L5
19.	How lexical analyser is generated with Lex? Show the process to detect sequence of input tokens with a diagram.	05	CO1/3,5	L3
20.	Provide design of simple lexical analyzer generator with a neat diagram.	08	CO1/3,5	L6
21.	Describe the structure of a LEX program and construct Lex program for detection of a few tokens such as, if, then, else, number etc.	06	CO1/3,5	L6

Unit II				
Sl. No.	**Questions**	**Marks**	**CO/PO**	**Bloom's Level**
1.	Explain the algorithms for FIRST and FOLLOW with suitable examples.	06	CO2/1, 2,3,5	L1
2.	Find the FIRST and FOLLOW sets for the grammar given below. E → TE' E'→ +TE'|ε T → FT' T'→ *FT'| ε F → (E)|id	06	CO2/1, 2,3,5	L5
3.	Find the FIRST and FOLLOW for the grammar, S→iEtSS'|a S'→eS| ε E→b	06	CO2/1, 2,3,5	L5
4.	Construct a LL (1) parsing table for the grammar E→TE' E'→+TE'| ε T'→FT' T'→*FT'| ε F→ (E) | id	10	CO2/1, 2,3,5	L6
5.	Show the model of a predictive parser and write the predictive parsing algorithm.	07	CO2/1, 2,3,5	L3
6.	Construct LL (1) parsing table for grammar and analyse the ambiguity situations. S→iEtSS'|a S'→eS| ε E→b	10	CO2/1, 2,3,5	L4,L6
7.	Explain shift reduce parser with grammar shown below for an input id*id+id and show the stack moves . E→E+E E→E*E E→ (E) E→id	10	CO2/1, 2,3,5	L2

8.	Find the FIRST and FOLLOW sets for the grammar, A→a|B|C B→b| ε C→ce|de	06	CO2/1, 2,3,5	L5
9.	Explain parser generator with a neat diagram.	05	CO2/1, 2,3,5	L2
10.	Compute canonical collection of sets of LR(0) items for grammar S→L=R|R, L→*R|id, R→L and show shift reduce conflicts in the items.	10	CO2/1, 2,3,5	L5
11.	Compute collection of sets of LR(0) items and construct the automaton &SLR parsing table for the grammar, E→E+T | T T→T*F | F E→(E) | id	12	CO2/1, 2,3,5	L6
12.	Show the stack moves of an LR parser on id*id+id for the grammar in problem 10. Choose LR(0) automaton or SLR table.	09	CO2/2, 3,5	L3
13.	Produce the algorithm for LR parsing and the algorithm for constructing SLR parsing table.	09	CO2/1, 2,3,5	
14.	Discuss YACC, its features and applications.	07	CO2/1, 2,3,5	L2
15.	Construct LR parsing table for grammar, S→iSeS|iS|a and show the parsing actions with stack moves on an input iiaea.	10	CO2/1, 2,3,5	L6
16.	Describe error recovery procedure in LR parser.	07	CO2/1, 2,3,5	L2
17.	Construct LR(1) set of items and show the automaton for the grammar, S→CC C→cC |d	10	CO2/1, 2,3,5	L6
18.	Construct canonical LR(1) parsing table for the grammar in problem 15 and identify ACTION and GOTO entries.	08	CO2/1, 2,3,5	L6
19.	Describe the procedure for constructing LR(1) sets of items, and show the steps.	10	CO2/1, 2,3,5	L2
20.	Construct LALR parsing table for the grammar in problem 15.	10	CO2/2,	L6

			3,5	
21.	Discuss and analyse various methods for construction of efficient LALR parsing tables.	08	CO2/1, 2,3,5	L2

Unit III (note: both * and × represent multiplication)

Sl. No.	Questions	Marks	CO/PO	Bloom's Level
1.	Explain syntax directed translation with an example.	07	CO3/2,3	L2
2.	Define inherited & synthesized attributes and justify their uses.	06	CO3/2,3	L5
3.	What is syntax directed definition (SDD) ? Construct SDD for simple desk calculator, showing production & semantic rules.	10	CO3/2,3	L6
4.	Construct parse tree for input string 3*5+4n, with SDD for the below given grammar , L → En E→E1 + T E →T T→T1* F T→F F→(E) | digit	10	CO3/2,3	L6
5.	Provide postfix translation scheme for implementing desk calculator.	08	CO3/2,3	L1
6.	Show how desk calculator can be implemented in stack.	08	CO3/2,3	L3
7.	Explain the process of intermediate code generation.	08	CO3/2,3	L2
8.	Justify the significance of construction of DAG in compiler design.	05	CO3/2,3	L5
9.	Represent a+ a × (b-c) + (b-c) × d in a syntax tree form and construct a triple for this.	06	CO3/2,3	L2
10.	Show the steps for constructing the DAG for a+ a × (b-c) + (b-c) × d.	05	CO3/2,3	L3
11.	Write and analyse the SDD and semantic rules to produce syntax tree .	06	CO3/2,3	L4
12.	Show a DAG and value number array for expression i= i + 10 .	06	CO3/2,3	L3

13.	Derive 3 address code for the expression, a = b × -c + b × -c and construct a quadruple for this expression.	08	CO3/2,3	L6
14.	Write 3 address code for a + a × (b - c) + (b - c) × d and construct a DAG and find quadruple, triple and indirect triple for this.	10	CO3/2,3	L6
15.	Show translation of array expression and analyse it.	10	CO3/2,3	L4
16.	Show annotated parse tree for c + a[i][j] and produce its 3 address representation.	06	CO3/2,3	L3
17.	Derive control flow translation of simple if statement.	08	CO3/2,3	L4
18.	Generate 3 address code for Boolean expressions and explain the procedure.	08	CO3/2,3	L5
19.	Show SDD for various flow of control statements.	07	CO3/2,3	L3
20.	Discuss backpatching and show how it is done for Boolean expressions.	10	CO3/2,3	L2
21.	Write an annotated parse tree for x < 100 || x > 200 && x != y , and analyse the way it is derived.	07	CO3/2,3	L4
22.	Describe translation of switch statements.	08	CO3/2,3	L2
23.	Illustrate intermediate code generation for procedures, with an example n = f(a[i])	10	CO3/2,3	L3

Unit IV

Sl. No.	Questions	Marks	CO/PO	Bloom's Level
1.	How is run time memory arranged in code and data area? Explain with a block diagram.	05	CO4/3,7	L2
2.	Explain stack allocation schemes.	10	CO4/3,7	L2
3.	Explain the issues in the design of code generator.	10	CO4/3,7	L2
4.	Show the structure of activation record and explain how it is	06	CO4/3,7	L3

	implemented with an example.			
5.	Show how variable length data is stored on stack with an example.	07	CO4/3,7	L3
6.	Provide the design of simple code generator using code generation algorithm and GetReg.	08	CO4/3,7	L6
7.	Write a simple code generator algorithm for the three address instruction X=Y+Z .	07	CO4/3,7	L1
8.	Translate the basic block consisting of 3 address statements below and show the register allocations. t=a-b u=a-c v =t+u a=d d=v+4	10	CO4/3,7	L6
9.	List all techniques for basic block generation and give an example for each of them.	10	CO4/3,7	L1
10.	Construct DAG for, X=A[i] a[j] =Y Z=a[i] And identify basic blocks for this.	08	CO4/3,7	L6
11.	Why is it important to do peephole optimization? Justify.	05	CO4/3,7	L5
12.	What are basic blocks and flow graphs? Write an algorithm for partitioning three address instructions into basic blocks.	08	CO4/3,7	L1

13.	Derive the intermediate code, draw the flow graph and find out the basic blocks for the code shown below.
	for i from 1 to 10 do
	for j from 1 to 10 do
	a[i,j] = 0.0;
	for i from 1 to 10 do
	a[i,i] = 1.0;
		10	CO4/3,7	L6

14.	Explain the optimal code generation for expression with examples.	06	CO4/3,7	L2
15.	Generate a tree labeled with Ershov numbers and produce a 3 address code for source code given below. t1=a-b t2=c+d t3=e*t2 t4=t1+t3	08	CO4/3,7	L6
16.	Analyse the working of peephole optimizations with specific examples.	10	CO4/3,7	L4

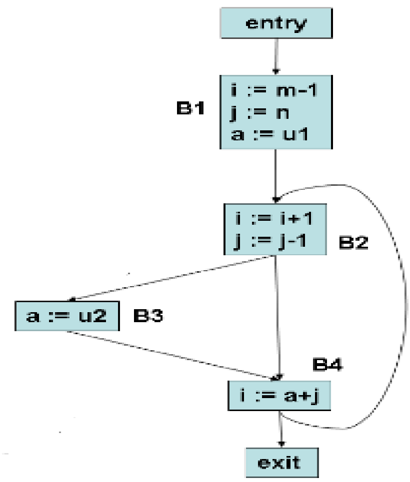
17.	For the following program segment generate 3 address code. Identify basic blocks and draw the flow graph.
	for i=1 to 20 do
	for j=1 to 20 do
	a[i,j]=10.0;
	for i=1 to 20 do
	a [i,i]=0.0;
		10	CO4/3,7	L6

18.	Compute the basic block for the following 3 address code and justify your answer. 1) PROD=0 2) i=1 3) t2=addr(A)-4 4) t4=addr(B)-4 5) t1=4*i 6) t3=t2[t1] 7) t5=t4[t1] 8) t6=t3*t5 9) PROD=PROD+t6 10) i=i+1 11) if (i<20) goto 5	06	CO4/3,7	L6
19.	Describe how register assignment and allocation is done in code generator.	08	CO4/3,7	L2
20.	Explain optimization of basic block using DAG representation. Construct a DAG for the code block shown below. a=b+c b=a-d	07	CO4/3,7	L6

	c=b+c d=a-d			
21.	Illustrate usage count method and live variable analysis for register allocation in a loop construct of a program.	06	CO4/3,7	L3

Unit V

Sl. No.	Questions	Marks	CO/PO	Bloom's Level
1.	Summarise various sources of code optimization.	07	CO5/1,3,5,6,7	L1
2.	Illustrate semantic preserving code transformations.	06	CO5/1,3,5,6,7	L3
3.	Analyze how optimization can be achieved with common subexpression elimination. Show an example.	06	CO5/1,3,5,6,7	L4
4.	Construct example for code optimization using copy propagation.	05	CO5/1,3,5,6,7	L6
5.	Justify how code motion can result in optimization giving an example.	06	CO5/1,3,5,6,7	L5
6.	How can strength reduction yield dead code? Explain with an example of a loop, justifying that strength reduction results in optimization.	05	CO5/1,3,5,6,7	L5
7.	Illustrate data flow abstraction with an example.	07	CO5/1,3,5,6,7	L2
8.	Discuss IN and OUT sets and state their purpose.	05	CO5/1,3,5,6,7	L1
9.	Comprehend the concept of reaching definitions with examples.	06	CO5/1,3,5,6,7	L2
10.	Show flow graph for illustrating reaching definition and explain.	05	CO5/1,3,5,6,7	L3
11.	Write iterative algorithms for reaching definitions.	06	CO5/1,3,5,6,7	L1
12.	Describe live variable analysis and justify its uses.	08	CO5/1,3,5,6,7	L5
13.	Determine the IN & OUT sets as well as USE and DEF sets for each of the basic blocks in the example given below.	08	CO5/3,5,6,7	L6

				
14.	Substantiate the working of constant propagation and strength reduction for code optimization with suitable examples.	06	CO5/1,3,5,6,7	L5
15.	Analyse partial redundancy elimination for code optimization.	06	CO5/1,3,5,6,7	L2
16.	Compare and analyse partial redundancy and full redundancy in programs.	04	CO5/1,3,5,6,7	L4
17.	With an example demonstrate the working of depth first analysis of a flow graph.	10	CO5/1,3,5,6,7	L2
18.	Outline the functioning of symbolic analysis of programs.	07	CO5/1,3,5,6,7	L2
19.	Elaborate power management optimization in programs.	10	CO5/1,3,5,6,7	L2
20.	Discuss green compilers and identify how that can be achieved.	10	CO5/1,3,5,6,7	L2
21.	Identify the features of region based analysis. Explain the algorithms for region based analysis.	10	CO5/1,3,5,6,7	L2
22.	Describe various tools for green compilation.	06	CO5/1,3,5,6,7	L2