



SRM VALLIAMMAI ENGINEERING COLLEGE
SRM Nagar, Kattankulathur – 603203.



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
QUESTION BANK

SUBJECT : CS8602 – COMPILER DESIGN

SEM / YEAR : VI/III

UNIT I -INTRODUCTION TO COMPILERS			
Structure of a compiler – Lexical Analysis – Role of Lexical Analyzer – Input Buffering – Specification of Tokens – Recognition of Tokens – Lex – Finite Automata – Regular Expressions to Automata – Minimizing DFA.			
PART-A (2 - MARKS)			
Q. No	QUESTIONS	Competence	BT Level
1.	Define tokens, patterns and lexemes.	Remember	BTL-1
2.	Classify approach would you use to recover the errors in lexical analysis phase.	Apply	BTL-3
3.	Apply the regular expression for identifier and white space.	Apply	BTL-3
4.	Point out why is buffering used in lexical analysis? What are the commonly used buffering methods?	Analyze	BTL-4
5.	Define transition diagram for an identifier.	Remember	BTL-1
6.	Compare syntax tree and parse tree.	Analyze	BTL-4
7.	Summarize the issues in a lexical analyzer.	Evaluate	BTL-5
8.	Define buffer pair.	Remember	BTL-1
9.	Differentiate the features of DFA and NFA.	Understand	BTL-2
10.	State the interactions between the lexical analyzer and the parser.	Remember	BTL-1
11.	Explain parse tree and construct a parse tree for $-(id + id)$	Evaluate	BTL-5
12.	Describe the operations on languages.	Remember	BTL-1
13.	List out the phases of a compiler.	Remember	BTL-1
14.	Generalizes the advantage of having sentinels at the end of each buffer halves in buffer pairs.	Create	BTL-6
15.	Classify the four software tools that generate parser.	Analyze	BTL-4
16.	Discuss Regular expression and the Algebraic properties of Regular Expression.	Understand	BTL-2
17.	Formulate the regular expressions are used though the lexical constructs of any programming language can be described using context free grammar.	Create	BTL-6
18.	Apply a grammar for branching statements.	Apply	BTL-3
19.	Express the main idea of NFA? And discuss with examples (a/b)*	Understand	BTL-2
20.	Define lex. Discuss the components of a lex.	Understand	BTL-2

PART-B (13- MARKS)																		
1.	Describe the various phases of compiler with suitable example	(13)	Remember BTL1															
2	(i)Give the structure of compiler (ii) Analyze structure of compiler with an assignment statement	(4) (9)	Analyze BTL4															
3.	(i). Discuss in detail about the role of Lexical analyzer with the possible error recovery actions. (ii)Describe in detail about issues in lexical analysis.	(7) (6)	Understand BTL2															
4	(i) Describe the Input buffering techniques in detail. (ii)Discuss how a finite automaton is used to represent tokens and perform lexical analysis with examples.	(7) (6)	Remember BTL1															
5	Summarize in detail about how the tokens are specified by the compiler with suitable example.	(13)	Understand BTL2															
6	Define Finite Automata. Differentiate Deterministic Finite Automata and Non-Deterministic Finite Automata with examples.	(13)	Understand BTL2															
7	(i) Solve the given regular expression $(a/b)^* abb (a/b)^*$ into NFA using Thompson construction. (ii).Compare NFA and DFA.	(7) (6)	Apply BTL3															
8	Create DFA the following NFA. $M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$ Where $\delta(q_0, 0) = \{q_0, q_1\}$ $\delta(q_0, 1) = \{q_1\}$ $\delta(q_1, 0) = \phi$ $\delta(q_1, 1) = \{q_0, q_1\}$	(13)	Create BTL6															
9	(i) Show how the DFA is directly converted from an augmented regular expression $((\epsilon/a)b^*)^*$. (ii)Draw NFA for the regular expression ab^*/ab	(7) (6)	Apply BTL3															
10.	(i) Define the language accepted by FA. Convert the following NFA into DFA. <table border="1" style="margin-left: 20px;"> <tr> <td></td> <td>0</td> <td>1</td> </tr> <tr> <td>$\rightarrow p$</td> <td>{p,q}</td> <td>{p}</td> </tr> <tr> <td>q</td> <td>{r}</td> <td>{r}</td> </tr> <tr> <td>r</td> <td>{s}</td> <td>ϕ</td> </tr> <tr> <td>*s</td> <td>{s}</td> <td>{s}</td> </tr> </table>		0	1	$\rightarrow p$	{p,q}	{p}	q	{r}	{r}	r	{s}	ϕ	*s	{s}	{s}	(7) (6)	Remember BTL1
	0	1																
$\rightarrow p$	{p,q}	{p}																
q	{r}	{r}																
r	{s}	ϕ																
*s	{s}	{s}																
11	Define Lex and Lex specifications. How lexical analyzer is constructed using lex? Give an example.	(13)	Remember BTL1															
12	(i) Explain an algorithm for Lex that recognizes the tokens. (ii) Describe in detail the tool for generating lexical analyzer.	(7) (6)	Evaluate BTL5															
13	(i) Analyze the algorithm for minimizing the number of states of a DFA	(7)	Analyze BTL4															

	i) Minimize DFA using Thompson Construction. (6) $(a/b)^* a (a/b) (a/b)$		
14 S	ow the minimized DFA for the regular expression: (13) $(0+1)^* (0+1) 1 0$. $ b)^* a (a b) (a b) (a b)$.	Apply	BTL3
PART-C (15- MARK)			
1.	(i) Create languages denoted by the following regular (9) expressions a) $(a b)^* a (a b) (a b)$ b) $a^* b a^* b a^* b a^*$ c) $!! (aa bb)^* ((ab ba)(aa bb)^* (ab ba)(aa bb)^*)^*$ (6) (ii) Write regular definitions for the following languages: a) All strings of lowercase letters that contain the five vowels in order. b) All strings of lowercase letters in which the letters are in ascending lexicographic order. c) Comments, consisting of a string surrounded by / and /, without an intervening */, unless it is inside double-quotes ("	Create	BTL6
2.	Find transition diagrams for the following regular expression (15) and regular definition. <ul style="list-style-type: none"> • $a(a b)^* a$ • $((\epsilon a)b^*)^*$ • All strings of digits with at most one repeated digit. • All strings of a's and b's that do not contain the substring abb. • All strings of a's and b's that do not contain the subsequence abb. 	Evaluate	BTL5
3.	(i) Prove that the following two regular expressions are equivalent (10) by showing that the minimum state DFA's are same : a) $(a/b)^*$ b) $(a^*/b^*)^*$ (ii) Minimize DFA using Thompson Construction (5) $(a/b)^* a b b (a/b)^*$	Analyze	BTL4
4.	Generalize and give an example one regular expression if we (15) to revise the definition of a DFA to allow zero or one transition out of each state on each input symbol. Some regular expressions would then have smaller DFA's than they do under the standard definition of a DFA. Give and generalize an example of one such regular expression.	Create	BTL6
UNIT II SYNTAX ANALYSIS			

Role of Parser – Grammars – Error Handling – Context-free grammars – Writing a grammar – Top Down Parsing - General Strategies Recursive Descent Parser Predictive Parser-LL(1) Parser-Shift Reduce Parser-LR Parser- LR (0)Item Construction of SLR Parsing Table - Introduction to LALR Parser - Error Handling and Recovery in Syntax Analyzer-YACC.

STUCOR APP

	$L \rightarrow L, S \mid S$ (ii) Analyze, Is it possible, by modifying the grammar in any way to construct predictive parser for the language of $S \rightarrow SS + \mid SS * \mid a$ string "aa+ a *". (5)		
4.	(i) Evaluate predictive parsing table and parse the string id+id*id. (7) And find FIRST and FOLLOW. $E \rightarrow E+T \mid T$ $T \rightarrow T * F \mid F$ $F \rightarrow (E) \mid id$ (ii) Construct Stack implementation of shift reduce parsing for the (6) grammar $E \rightarrow E+E$ $E \rightarrow E * E$ $E \rightarrow (E)$ $E \rightarrow id$ and the input string id1+id2*id3	Evaluate	BTL5
5.	(i). Describe on detail about the role of parser. (7) (ii) Discuss about the context-free grammar. (6)	Remember	BTL1
6.	(i). What are the different kinds of syntax error phased by a program? Explain in detail. (7) (ii). What are the Error recovery techniques used in Predictive parsing? Explain in detail. (6)	Remember	BTL1
7.	Give the predictive parser for the following grammar. $S \rightarrow (L) \mid a$ $L \rightarrow L, S \mid S$ i. Give a rightmost derivation for (a, (a, a)) and show the handle of each right-sentential form. (5) ii. Show the steps of a shift reduce parser. (8)	Understand	BTL2
8.	Analyze the following grammar is a LR(1) grammar and construct (13) LALR parsing table. $S \rightarrow Aa \mid bAc \mid dC \mid bda$ $A \rightarrow d$. Parse the input string bdc using the table generated.	Analyze	BTL4
9.	(i) Define YACC parser generator. List out the Error recovery actions in YACC. (8) (ii) List out different error recovery strategies. Explain them. (5)	Remember	BTL1
10.	(i) Show SLR parsing table for the following grammar $S \rightarrow Aa \mid bAc \mid Bc \mid bBa$ $A \rightarrow d$ $B \rightarrow d$ And parse the sentence "bdc" and "dd". (8) (ii) Construct a parse tree for the input string w-cad using top down	Apply	BTL-3

	<p>parser . $S \rightarrow cA$ d $A \rightarrow ab a$</p>	(5)		
11.	<p>(i) Define SLR (1) parser. Describe the Steps for the SLR parser.</p> <p>(ii) Predict the following grammar for generate the SLR parsing table.</p> $E \rightarrow E+T \mid T$ $T \rightarrow T * F \mid F$ $F \rightarrow F * a b$	(5) (8)	Understand	BTL2
12	<p>(i) Consider the following grammar $S \rightarrow AS b$ $A \rightarrow SA a$. Construct the SLR parse table for the grammar.</p> <p>(ii) Show the actions of the parser for the input string “abab”.</p>	(10) (3)	Apply	BTL3
13.	<p>Give the LALR for the given grammar. $E \rightarrow E + T \mid T$, $T \rightarrow T * F \mid F$, $F \rightarrow (E) / id$ and parse the following. $(a + b) * c$</p>	(13)	Understand	BTL-2
14.	<p>Examine the following grammar using canonical parsing table.</p> $E \rightarrow E + T$ $F \rightarrow (E)$ $E \rightarrow T$ $F \rightarrow id.$ $T \rightarrow T * F$ $T \rightarrow F$	(13)	Remember	BTL-1

PART-C (15 - MARKS)

1.	<p>What is an operator grammar? Draw the precedence function graph for the following table.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td></td> <td>a</td> <td>(</td> <td>)</td> <td>,</td> <td>\$</td> </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>		a	()	,	\$	a			>	>	>	(<	<	=	<)			>	>	>	,	<	<	>	>		\$	<	<				(15)	Create	BTL6
	a	()	,	\$																																			
a			>	>	>																																			
(<	<	=	<																																				
)			>	>	>																																			
,	<	<	>	>																																				
\$	<	<																																						
PART-A (2 - MARKS)																																								
Q.No	QUESTIONS																																							

	1. Write the rule to eliminate left recursion in a grammar. Prepare and Eliminate the left recursion for the grammar. $S \rightarrow Aa \mid b$ $A \rightarrow Ac \mid Sd \mid \epsilon$		
	2. Define handle pruning.		
	3. Solve FIRST and FOLLOW by use the LL(1) grammar.		
	4. List the concepts of Predictive parsing and shift reduce pars		
	5. Differentiate Top Down parsing and Bottom Up parsing.		
	6. Define Recursive Descent Parsing.		
	7. List out the properties of parse tree.		
	8. Compare and contrast top down parsing with bottom up pa techniques.		
	9. Solve the following grammar is ambiguous: $S \rightarrow aSbS \mid bSa$		
	10. Define kernel and non-kernel items.		
	11. Difference between ambiguous and unambiguous grammar		
	12. Define parser. Give the advantages and disadvantages of LL parsing.		
	13. Define Phrase level error recovery.		
	14. Evaluate the conflicts encountered while parsing.		
	15. Analyze the categories of shift reduce parsing.		
	16. How to create an input and output translator with YACC?		
	17. Summarize the Error recovery scheme in yacc.		
	18. What is the main idea of Left factoring? Give an example.		
	19. Discuss when Dangling reference occur?		
	20. Examine the approach would you use in Panic mode error recovery.		
	PART-B (13- MARKS)		
	1. (i) Explain left recursion and Left Factoring. (ii) Eliminate left recursion and left factoring for the following grammar. $E \rightarrow E + T \mid E - T \mid T$ $T \rightarrow a \mid b \mid (E) .$		
	2. (i) What is an ambiguous and unambiguous grammar? Identify the following grammar is ambiguous or not. $E \rightarrow E + E \mid E * E \mid (E) \mid - E \mid id$ for the sentence $id + id * id$ (ii) Prepare the following grammar is LL(1) but no SLR(1). $S \rightarrow AaAb \mid BbBa$ $A \rightarrow \epsilon$ $B \rightarrow \epsilon$		
	3. (i) Illustrate the predictive parser for the following grammar $S \rightarrow (L) \mid a$		
2	Explain in detail about the various types of Top –down parsing. (15)	Evaluate	BTL5

3	Analyze the LR parsing algorithm with an example (15)	Analyze	BTL4
4	What is CFG . Explain in detail about the Context-Free Grammar. (15)	Evaluate	BTL5
UNIT-III INTERMEDIATE CODE GENERATION			
Syntax Directed Definitions, Evaluation Orders for Syntax Directed Definitions, Intermediate Languages: Syntax Tree, Three Address Code, Types and Declarations, Translation of Expressions, Type Checking.			
PART-A (2 - MARKS)			
1.	List out the two rules for type checking.	Remember	BTL1
2.	Compare synthesized attributes and inherited attributes.	Analyze	BTL4
3.	What is Annotated parse tree?	Remember	BTL1
4.	Define Type checker.	Remember	BTL1
5.	What is a syntax tree? Draw the syntax tree for the assignment statement $a := b * -c + b * -c$	Create	BTL6
6.	Define type systems.	Remember	BTL1
7.	Express the rule for checking the type of a function.	Understand	BTL2
8.	Define Syntax directed definition of a simple desk calculator.	Remember	BTL1
9.	Summarize about the S-attributed definition?	Evaluate	BTL5
10.	Give the difference between syntax-directed definitions and translation schemes.	Understand	BTL2
11.	State the type expressions.	Remember	BTL1
12.	Illustrate the methods of implementing three-address statements.	Apply	BTL3
13.	Differentiate S-attribute and L-attribute definitions.	Analyze	BTL4
14.	Create the target machine instructions to implement the call statement in static allocation.	Create	BTL6
15.	Translate the conditional statement if $a < b$ then 1 else 0 into three address code.	Understand	BTL2
16.	Test whether the following rules are L-attribute or not? Semantic rules $A.s = B.b;$ $B.i = f(C.c, A.s)$	Evaluate	BTL5
17.	What are the methods of representing a syntax tree?	Understand	BTL2
18.	Explain the syntax directed definition for if-else statement	Analyze	BTL4
19.	Examine the usage of syntax directed definition	Apply	BTL3
20.	Discuss the three address code sequence for the assignment statement. $d = (a-b) + (a-c) + (a-c)$	Understand	BTL2
PART-B (13- MARKS)			
1.	Discuss the following in detail about the Syntax Directed Definitions. (i) Inherited Attributes and Synthesized attributes. (7) (ii) Evaluate SDD of a parse tree. (6)	Understand	BTL2

2.	Evaluate the expressions for the SDD annotated parse tree for the following expressions. (i) $3 * 5 + 4n$ (7)	Evaluate	BTL5
----	---	----------	------

	(ii) $1 * 2 * 3 * (4 + 5)$ (6)		
3.	<p>Suppose that we have a production $A \rightarrow BCD$. Each of the four non-terminal A, B, C and D have two attributes: S is a synthesized attribute and i is an inherited attribute. Analyze For each of the sets of rules below tell whether (i) the rules are consistent with an S-attributed definition (ii) the rules are consistent with an L-attributed definition and (iii) whether the rules are consistent with any evaluation order at all?</p> <p>a) $A.s = B.i + C.s$ b) $A.s = B.i + C.s$ and $D.i = A.i + B.s$.</p>	Analyze	BTL4
4.	<p>Apply the S-attributed definition and constructs syntax trees for a simple expression grammar involving only the binary operators $+$ and $-$. As usual, these operators are at the same precedence level and are jointly left associative. All nonterminal have one synthesized attribute node, which represents a node of the syntax tree. Production: $L \rightarrow E\\$ $E \rightarrow E_1 + T, E \rightarrow T, T \rightarrow T_1 * F, T \rightarrow (E), T \rightarrow \text{digit}$.</p>	Apply	BTL3
5.	<p>Discuss in detail about (i) Dependency graph (10) (ii) Ordering Evaluation of Attributes. (3)</p>	Understand	BTL2
6.	<p>Create variants of Syntax tree. Explain in detail about it with suitable examples. (13)</p>	Create	BTL6
7.	<p>(i). Analyze the common three address instruction forms. (7) (ii). Explain the two ways of assigning labels to the following three address statements (6) Do $i = i + 1;$ While $(a[i] < v);$</p>	Analyze	BTL4
8.	<p>Describe in detail about (i) Quadruples (7) (ii) Triples. (6)</p>	Remember	BTL1
9.	<p>(i) Describe in detail about addressing array Elements. (6) (ii) Discuss in detail about Translation of array reference. (7)</p>	Remember	BTL1
10.	<p>Describe in detail about types and declaration with suitable examples. (13)</p>	Remember	BTL1
11.	<p>Compare three address code for expression with the Incremental translation. (13)</p>	Analyze	BTL-4
12.	<p>Show the intermediate code for the following code segment along with the required syntax directed translation scheme (13) while ($i < 10$) if ($i \% 2 == 0$) evensum = evensum + i else</p>	Understand	BTL-2

	oddsun = oddsun + i		
13.	(i) State the rules for type checking with example. (7) (ii) Give an algorithm for type inference and polymorphic function. (6)	Remember	BTL-1
14.	Illustrate an algorithm for unification with its operation. (13)	Apply	BTL-3
PART-C(15 -MARKS)			
1.	Create the following und the arithmetic expression $a + (b+c)^*$ (15) into (i) Syntax tree (ii) Quadruples (iii) Triples (iv) Indirect Triples	Create	BTL-6
2.	Explain the steps for constructing a DAG. Construct the DAG for (15) the following expression $((x+y) - ((x+y)*(x-y))) + ((x+y)*(x-y))$	Evaluate	BTL5
3.	Generate an intermediate code for the following code segment (15) with the required syntax-directed translation scheme. if (a > b) x = a + b else x = a - b	Create	BTL-6
4.	What is Type conversion? What are the two types of type (15) conversion? Evaluate the rules for the type conversion.	Evaluate	BTL5
UNIT IV- RUN-TIME ENVIRONMENT AND CODE GENERATION			
Storage Organization, Stack Allocation Space, Access to Non-local Data on the Stack, Heap Management - Issues in Code Generation - Design of a simple Code Generator.			
PART-A (2 -MARKS)			
1.	List out limitations of the static memory allocation.	Remember	BTL1
2.	How the storage organization for the run-time memory is organized?	Apply	BTL3
3.	What is heap allocation?	Remember	BTL1
4.	How the activation record is pushed onto the stack?	Apply	BTL3
5.	Analyze the storage allocation strategies.	Analyze	BTL4
6.	State the principles for designing calling sequences.	Remember	BTL1
7.	List out the dynamic storage techniques.	Remember	BTL1
8.	Define the non-local data on stack.	Remember	BTL1
9.	Define variable data length on the stack.	Remember	BTL1
10.	Differentiate between stack and Heap allocation	Analyze	BTL4
11.	Distinguish between static and dynamic storage allocation.	Understand	BTL2
12.	Discuss the main idea of Activation tree.	Understand	BTL2
13.	Give the fields in an Activation record.	Understand	BTL2
14.	Compose space efficiency and program efficiency.	Create	BTL6
15.	Construct typical memory hierarchy configuration of a computer.	Evaluate	BTL5

16.	How would you solve the issues in the design of code generators?	Apply	BTL3
-----	---	-------	------

17.	Evaluate Best-fit and Next-fit object placement.	Evaluate	BTL5
18.	Prepare optimal code sequence for the given sequence t=a+b t=t*c t=t/d	Create	BTL6
19.	Analyze the different forms of machine instructions.	Analyze	BTL4
20.	Discuss the four principle uses of registers in code generation.	Understand	BTL2
PART-B (13- MARKS)			
1.	(i) Illustrate the storage organization memory in the perspective of (8) compiler writer with neat diagram. (ii) Compare static versus dynamic memory allocation. (5)	Apply	BTL3
2.	Explain in detail about the various issues in code generation with examples. (13)	Evaluate	BTL5
3.	(i) Develop a quicksort algorithm for reads nine integers into an array a and sorts them by using the concepts of activation tree. (ii) Give the structure of the action record. (4)	Create	BTL6
4.	How to a design a call sequences and analyze the principles of activation records with an example. (13)	Analyze	BTL4
5.	Discuss in detail about the activation tree and activation record with suitable example (13)	Understand	BTL 2
6.	(i) Analyze the data access without nested procedure and the issues (7) with nested procedure. (ii) Give the version of quicksort in ML style using nested procedure. (6)	Analyze	BTL4
7.	(i) Discuss in detail about heap manager. (7) (ii) Describe in detail about the memory hierarchy of a computer (6)	Understand	BTL2
8.	Define fragmentation? Describe in detail about how to reduce the fragment. (13)	Remember	BTL1
9.	Write short notes on the following i. Best fit and next object placement. (7) ii. Managing and coalescing free space (6)	Remember	BTL1
10.	Examine the problems with manual deallocation of memory and explain how the conventional tools are used to cope with the complexity in managing memory. (13)	Remember	BTL1
11.	Explain in detail about instruction selection and register allocation of code generation. (13)	Analyze	BTL4
12.	Illustrate in detail about the code generation algorithm with an example. (13)	Apply	BTL-3
13.	Describe the usage of stack in the memory allocation and discuss in detail about stack allocation space of memory. (13)	Understand	BTL-2
14.	Define the heap management of memory and describe in detail about it . (13)	Remember	BTL-1
PART-C (15-MARKS)			

1.	Suppose the heap consists of seven chunks, starting at address 0. The sizes of the chunks, in order, are 80, 30, 60, 50, 70, 20, 40	(15)	Evaluate	BTL-5
----	---	------	----------	-------

	bytes. When we place an object in a chunk, we put it at the high end if there is enough space remaining to form a smaller chunk (so that the smaller chunk can easily remain on the linked list of free space). However, we cannot tolerate chunks of fewer than 8 bytes, so if an object is almost as large as the selected chunk, we give it the entire chunk and place the object at the low end of the chunk. If we request space for objects of the following sizes: 32, 64, 48, 16, in that order, what does the free space list look like after satisfying the requests, if the method of selecting chunks is a) First fit. b) Best fit.		
2.	Compare the stack and heap allocation memory in detail with (15) suitable examples.	Analyze	BTL4
3.	Generate code for the following sequence assuming that n is in a memory location (15) s=0 i=0 L1 : if I > n goto L2 s=s+i i=i+1 goto L1 L2 :	Create	BTL-6
4.	Create following assignment statement into three address code (15) $D := (a-b) * (a-c) + (a-c)$ Apply code generation algorithm to generate a code sequence for the three address statement. (13)	Create	BTL-6

UNIT V- CODE OPTIMIZATION

Principal Sources of Optimization – Peep-hole optimization - DAG- Optimization of Basic Blocks
Global Data Flow Analysis - Efficient Data Flow Algorithm.

PART-A (2 -MARKS)

1.	List out the examples of function preserving transformations.	Remember	BTL1
2.	Illustrate the concepts of copy propagation.	Apply	BTL3
3.	State the use of machine Idioms.	Remember	BTL1
4.	Show the flow graph for the quicksort algorithm	Apply	BTL3
5.	Apply the basic block concepts, how would you representing the dummy blocks with no statements indicated in global dataflow analysis?	Apply	BTL3
6.	Identify the constructs for optimization in basic block.	Remember	BTL1
7.	List out the properties of optimizing compilers.	Remember	BTL1
8.	Define the term data flow analysis.	Remember	BTL1
9.	How is liveness of a variable calculated? Identify it.	Remember	BTL1
10.	What is DAG? Point out advantages of DAG.	Analyze	BTL4
11.	Give the uses of gen and Kill functions	Understand	BTL2
12.	Discuss the concepts of basic blocks and flow graphs.	Understand	BTL2
13.	Give the main idea of dead code elimination and constant folding.	Understand	BTL2

14.	Prepare the three address code sequence for the assignment statement.	Create	BTL6
-----	--	--------	------

	$d := (a - b) + (a - c) + (a - c).$		
15.	Construct and explain the DAG for the follow basic block. $d := b * c$ $e := a + b$ $b := b * c$ $a := e - d.$	Evaluate	BTL5
16.	What role does the target machine play on the code generation phase of the compiler? Analyze it.	Analyze	BTL4
17.	Draw the DAG for the statement $a = (a * b + c) - (a * b + c)$ and evaluate it.	Evaluate	BTL5
18.	Develop the code for the follow C statement assuming three registers are available. $x = a / (b + c) - d * (e + f)$	Create	BTL6
19.	Point out the characteristics of peephole optimization.	Analyze	BTL4
20.	Define algebraic transformations. Give an example	Understand	BTL2
PART-B(13 MARKS)			
1.	Explain briefly about the principal sources of optimization. (13)	Evaluate	BTL5
2.	(i). Explain in detail about optimization of basic blocks. (5) (ii).Construct the DAG for the following Basic block & explain it. (8) 1. $t1 := 4 * i$ 2. $t2 := a [t1]$ 3. $t3 := 4 * i$ 4. $t4 := b [t3]$ 5. $t5 := t2 * t4$ 6. $t6 := Prod + t5$ 7. Pro $d := t6$ 8. $t7 := i + 1$ 9. $i := t7$ 10. if $i \leq 20$ goto (1).	Analyze	BTL4
3.	Discuss the following in detail (i) Semantic preserving transformation (7) (ii) Global Common subexpression (6)	Understand	BTL2
4.	Write about the following in detail (5) (i) copy propagation (5) (ii) Dead code Elimination (3) (iii) code motion	Remember	BTL1
5.	Explain in detail about the data-flow schemas on basic block and (13) the transfer equations for reaching definitions with example	Analyze	BTL4
6.	(i) Illustrate the Iterative algorithm for reaching definitions (7) (ii) Discuss the live variable analysis (6)	Apply	BTL3
7.	Analyze Peephole optimization with suitable examples. (13)	Analyze	BTL4
8.	Demonstrate optimization of Basic Blocks with an example. (13)	Apply	BTL3

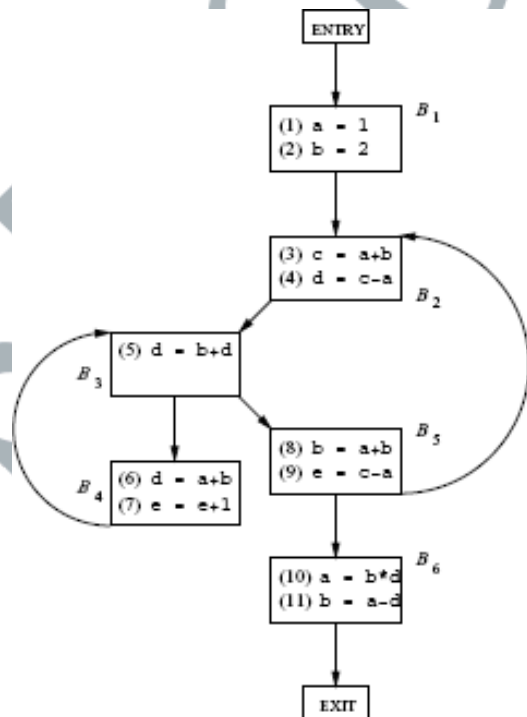
9.	(i) Discuss in detail about how to find Local Common Sub (8) expressions. (ii) Discuss in detail about the Use of Algebraic Identities. (5)	Understand	BTL2
----	--	------------	------

10.	Describe in detail about the flow of control optimization. Identify the methods to eliminate the unreachable code, load and store data.	(7) (6)	Remember	BTL1
11.	(i) Give an example to identify the dead code in the DAG. (ii) Describe the representation of array using DAG with example.	(5) (8)	Remember	BTL1
12.	Summarize in detail about the dataflow analysis of available expression with suitable example.	(13)	Understand	BTL2
13.	(i) Formulate steps to identify the loops in the basic block. (ii) Describe about induction variable and end reduction in strength	(7) (6)	Create	BTL6
14.	Describe the efficient data flow algorithms in detail.	(13)	Remember	BTL1

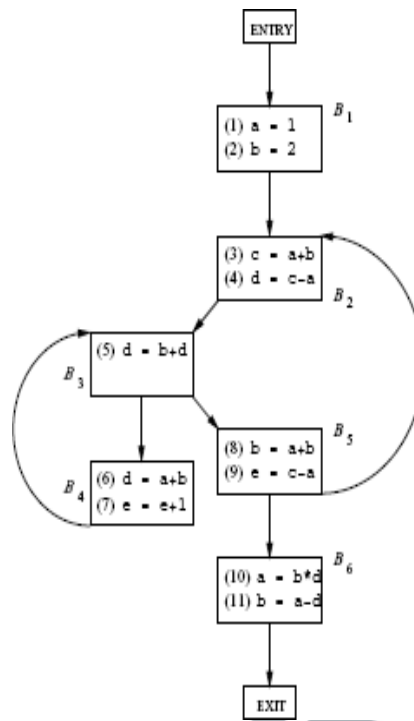
PART-C(15 MARKS)

1.	Create DAG and three – address code for the following C (15) program. (15) <pre> i = 1; s = 0; while (i<= 10) { s = s+ a[i] [i]; i = i + 1; } </pre>		Create	BTL6
----	---	--	--------	------

2.	Identify the loops of the flow graph, Identify the global common sub expression for each loop, Identify Induction variables for each loop and Identify loop invariant computation for each loop from the given diagram,	(15)	Create	BTL6
----	---	------	--------	------



3.	Compute the grn and Kill sets for each Block, In and Out sets for each block, Compute e_gen and e_kill from the given diagram.	(15)	Evaluate	BTL5
----	--	------	----------	------



4. **Analyze** the available expressions on the following code by (15) converting into basic blocks and compute global common sub-expression elimination. (15)

```

i = 0
a:= n-3
if i < a then loop else end
label loop
b:= i -4
c:= p + b
d:= M[c]
e:=d-2
f:=i-4
g:=p+f
m[g]:=e
i:=i+1
a:=n-3
if i < a then loop else end
label end
  
```

Analyze BTL4