.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
QUESTION BANK

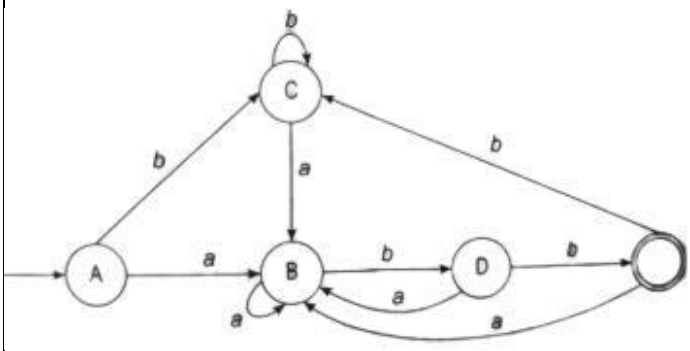
SUBJECT CODE : 19CS502

NAME: AUTOMATA THEORY AND

COMPILER DESIGN

SEM / YEAR : V/III

UNIT I -INTRODUCTION TO COMPILERS			
Phases of a compiler – Lexical Analysis – Role of Lexical Analyzer — Lex – Finite Automata – Regular Expressions to Automata –NFA to DFA- Minimizing DFA.			
PART-A (2 - MARKS)			
Q. No	**QUESTIONS**	**Competence**	**BT Level**
1.	Define tokens, patterns and lexemes.	Remember	BTL1
2.	Classify approach would you use to recover the errors in lexical analysis phase?	Apply	BTL3
3.	Apply the regular expression for identifier and white space.	Apply	BTL3
4.	Point out why is buffering used in lexical analysis?	Analyze	BTL4
5.	Define transition diagram for an identifier.	Remember	BTL1
6.	Compare syntax tree and parse tree.	Analyze	BTL4
7.	Summarize the issues in a lexical analyzer.	Evaluate	BTL5
8.	Define buffer pair.	Remember	BTL1
9.	Differentiate the features of DFA and NFA.	Understand	BTL2
10.	Identify the interactions between the lexical analyzer and the parser.	Remember	BTL1
11	State parse tree and construct a parse tree for –(id + id)	Evaluate	BTL5
12.	Name the operations on languages.	Remember	BTL1
13.	List out the phases of a compiler.	Remember	BTL1
14.	Generalizes the advantage of having sentinels at the end of each buffer halves in buffer pairs.	Create	BTL6
15.	Analyze and identify the symbol table for the following statements. int a,b; float c; char z;	Analyze	BTL4
16.	Discuss Regular expression and the Algebraic properties of Regular Expression.	Understand	BTL2
17.	Develop the Structure of lex program.	Create	BTL6
18.	Apply a grammar for branching statements.	Apply	BTL3
19.	Express the main idea of NFA? And discuss with examples (a/b)*	Understand	BTL2
20.	Define lex and give its execution steps.	Understand	BTL2

21	Differentiate interpreters and compilers	Analyze	BTL4
22	Apply the parse tree for the statement z:= x+y*130.	Apply	BTL3
23	Outline the role of lexical analysis in compiler design.	Understand	BTL2
24	Criticize the use of Input Buffering with simple examples.	Evaluate	BTL5
	PART-B (13- MARKS)		
1.	Describe the various phases of compiler with suitable example (13)	Remember	BTL1
2	(i)Give the structure of compiler. (4) (ii)Analyze structure of compiler with an assignment statement (9)	Analyze	BTL4
3.	(i).Discuss in detail about the role of Lexical analyzer with the possible error recovery schemes. (7) (ii)Describe in detail about issues in lexical analysis. (6)	Understand	BTL2
4	(i)Describe the Input buffering techniques in detail. (7) (ii)Discuss about the recognition of tokens with example (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	Solve the given regular expression into NFA using Thompson construction i)(a/b)* abb (a/b)*. (7) ii)ab*/ab (6)	Apply	BTL3
8	Create DFA the following regular expression.((ε /a)b*)* (13)	Create	BTL6
9	(i)Illustrate the algorithm for minimizing the number of states of a DFA (8) (ii)Minimize the following states of DFA (5) 	Apply	BTL3
10.	Describe in detail about the subset construction of DFA from NFA (13)	Remember	BTL1
11	Define Lex and Lex specifications. How lexical analyzer is constructed using lex? Give an example. (13)	Remember	BTL1
12	(i)Explain the lex program for tokens. (7) (ii) Describe in detail the tool for generating lexical analyzer. (6)	Evaluate	BTL5
13	Find the NFA for the given regular expression and find the (13)	Analyze	BTL4

	minimized DFA for the constructed NFA..(a* / b*)*			
14	Find the minimized DFA for the regular expression: (0 + 1) * (0 + 1) 1 0.	(13)	Analyze	BTL4
15	Discuss in detail about the output of each phase of compiler for the expression a:=b+c*50.	(13)	Understand	BTL2
16	Demonstrate the role of lexical analyzer in detail with necessary diagrams	(13)	Apply	BTL3
17	Determine the minimum -state DFA for the regular expression (a / b)* a (a/b)	(13)	Evaluate	BTL5
PART-C (15- MARK)				
1.	(i) Create languages denoted by the following regular expressions a) (a|b)*a(a|b)(a|b) b) a*ba*ba*ba* c) !! (aa|bb)*((ab|ba)(aa|bb)*(ab|ba)(aa|bb)*)* (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 (")	(9) (6)	Create	BTL6
2.	Find transition diagrams for the following regular expression and regular definition. a(a|b)*a ((ε|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.	(15)	Evaluate	BTL5
3.	Evaluate that the following two regular expressions are equivalent by showing that the minimum state DFA's are same (a / b) * (a * / b *) *	(15)	Evaluate	BTL5
4.	Explain in detail the tool for generating Lexical-Analyzer with an example program.	(15)	Evaluate	BTL5
5	Develop the Lex Program to recognize the identifiers, constants and operators	(15)	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.				

PART-A (2 - MARKS)			
1.	Eliminate the left recursion for the grammar. S → Aa | b A →Ac | Sd |ε	Create	BTL6
2.	Define handle pruning.	Remember	BTL1
3.	Compute FIRST and FOLLOW for the following grammar S → AS S →b A →SA A → a	Apply	BTL3
4.	State the concepts of Predictive parsing .	Remember	BTL1
5.	Differentiate Top Down parsing and Bottom Up parsing?	Understand	BTL2
6.	Define Recursive Descent Parsing.	Remember	BTL1
7.	State the different error recovery methods of predictive parsing.	Remember	BTL1
8.	Write an algorithm for finding FOLLOW.	Analyze	BTL4
9.	What is the main idea of Left factoring? Give an example.	Understand	BTL2
10.	Define LL(1) Grammar.	Remember	BTL1
11.	Difference between ambiguous and unambiguous grammar.	Analyze	BTL4
12.	Define parser. Explain the advantages and disadvantages of LR parsing?	Evaluate	BTL5
13.	Define Augmented Grammar with an example.	Remember	BTL1
14.	Evaluate the conflicts encountered while parsing?	Evaluate	BTL5
15.	Point out the categories of shift reduce parsing.	Analyze	BTL4
16.	How to create an input and output translator with YACC.	Create	BTL6
17.	Give the four possible actions of LR Parsing.	Understand	BTL2
18.	Solve the following grammar is ambiguous: S→aSbS / bSaS / €	Apply	BTL3
19.	Discuss when Dangling reference occur?	Understand	BTL2
20.	Illustrate the use of GOTO function.	Apply	BTL3
21.	Give the comparison between various LR parsers	Evaluate	BTL5
22.	Write down the structure of YACC file	Analyze	BTL4
23.	Differentiate Lex and yacc	Understand	BTL2
24.	Write about Closure Operation	Apply	BTL3
PART-B (13- MARKS)			
1.	(i)Explain left recursion and Left Factoring. (7) (ii)Eliminate left recursion and left factoring for the following grammar. (6) E → E + T | E - T | T T → a | b | (E).	Analyze	BTL4
2.	(i) Parse the input string 000111 for the grammar S-> 0S1| 01 (6) (ii) Construct a parse tree for the input string w-cad using top down parser . (7)	Create	BTL6

	S->cAd A->ab|a			
3.	(i)Analyze the give grammar to construct predictive parser S→ +SS | *SS | a with the string "+*aaa.	(13)	Analyze	BTL4
4.	(i) Evaluate predictive parsing table for the following grammar E→E+T | T T→T*F | F F→(E) | id (ii) Parse the string id+id*id	(9) (4)	Evaluate	BTL5
5.	Solve the following grammar for the predictive parser and parse the string 000111 S>0S1 S->01	(13)	Analyze	BTL2
6.	(i).Describe on detail about the various types of parser (ii)Discuss about the context-free grammar.	(7) (6)	Remember	BTL1
7.	(i).Discuss in detail aabout the role of parser. (ii).What are the Error recovery techniques used in Predictive parsing? Explain in detail.	(7) (6)	Remember	BTL1
8.	(i) Give the predictive parser table for the following grammar. S→ (L) | a L→ L, S | S (ii) Parse the string (a, (a, a)) .	(8) (5)	Understand	BTL2
9.	(i_Analyze the following grammar is a LALR grammar. S->CC C->cC|d (ii)Parse the input string ba using the table generated.	(13)	Analyze	BTL4
10.	(i)Define YACC parser generator. List out the Error recovery actions in YACC. (ii) Define SLR (1) parser. Describe the Steps for the SLR parser.	(8) (5)	Remember	BTL1
11	(i)Show SLR parsing table for the following grammar A->(A)|a ii)Differentiate SLR and CLR	(9) (4)	Apply	BTL3
12.	Solve the following grammar to generate the SLR parsing table. E→E+T | T T→T*F | F F→F*| a| b	(13)	Understand	BTL2
13.	(i) Consider the following grammar S →AS|b A→SA|a. Construct the SLR parse table for the grammar. (ii) Show the actions of the parser for the input string "abab".	(10) (3)	Apply	BTL3
14.	Give the LALR for the given grammar. S->AA A->Aa|b	(13)	Understand	BTL2

15.	Examine the following grammar using canonical parsing table. (13) S->CC C->cC|d	Remember	BTL1
16.	Explain SLR parser.Construct SLR parse for the given grammar. (13) S->L=R S->R L->*R L->id R->L	Evaluate	BTL5
17.	Show the bottom up parser for the following (13) The input aaa*a++ for the grammar S->SS+ S->SS* S->a	Apply	BTL3
	PART-C (15 -MARKS)		
1.	(i) What is Leftmost derivation and Rightmost derivation . (8) Draw leftmost derivation and Rightmost derivation for the following. E->E+E|E*E| id (ii) What is an ambiguous and unambiguous grammar? Identify (7) the following grammar is ambiguous or not. E→E+E | E*E | (E)|-E |id for the sentence id +id*id	Create	BTL6
2	Explain in detail about the various types of Top –down(15) parsing.	Evaluate	BTL5
3	Evaluate the LR parsing algorithm with an example (15)	Evaluate	BTL5
4	(i) What is CFG .Explain in detail about the Context-Free (8) Grammar (ii) Construct Stack implementation of shift reduce parsing for (7) the grammar E->E+E E->E*E E->(E) E->id and the input string id1+id2*id3 .	Evaluate	BTL5
5.	Discuss in detail about YACC Paser -Generator with an(15) example program	Create	BTL6
	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.	Identify the different types of intermediate representation.	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 postfix notation for the given expression a+b*c.	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.	Constrct the syntax directed definition for if-else statement	Analyze	BTL4
19.	Examine the usage of syntax directed definition	Apply	BTL3
20.	Show the three address code sequence for the assignment statement. d=(a-b)+(a-c)+(a-c)	Apply	BTL3
21.	Give the evaluation order of a SDD	Evaluate	BTL5
22	What is translation scheme?	Understand	BTL2
23.	How will you evaluate semantic rules?	Analyze	BTL4
24.	Illustrate how to construct syntax tree for an expression	Apply	BTL3
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.	Identify the annotated parse tree for the following expression (i)(3+4)*(5+6)n (6) (ii)1*2*3*(4+5)n (7) Using the given SDD Production — Semantic Rules D —>TL — L.inh = T.type T —> int — T.type =integer T —> float — T.type = float L —> L1, id — L1.inh = L.inh addType (id.entry, Linh)	Evaluate	BTL5
3.	Suppose that we have a production A→BCD. Each of the four non terminal A, B, C and D have two attributes: S is a (13)	Analyze	BTL4

	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? A.s = B.i + C.s A.s = B.i + C.s and D.i = A.i + B.s.			
4.	Illustrate in detail about the various instructions forms of three address instruction with suitable examples	(13)	Apply	BTL3
5.	Discuss in detail about (i)Dependency graph (ii)Ordering Evaluation of Attributes.	 (10) (3)	Understand	BTL2
6.	Create variants of Syntax tree. Explain in detail about it with suitable examples.	(13)	Create	BTL6
7.	(i).Analyse the common three address instruction forms. (ii). Explain the two ways of assigning labels to the following three address statements Do i=i+1; While (a[i]<v);	(7) (6)	Analyze	BTL4
8.	Describe.in detail about (i) Quadruples (ii) Triples.	 (7) (6)	Remember	BTL1
9.	(i) Describe in detail about addressing array Elements. (ii) Discuss in detail about Translation of array reference.	(6) (7)	Remember	BTL1
10.	Describe in detail about types and declaration with suitable examples.	(13)	Remember	BTL1
11.	Compare three address code for expression with the Incremental translation.	(13)	Analyze	BTL4

12.	Show the intermediate code for the following code segment along with the required syntax directed translation scheme
	while (i < 10)
	if (i % 2 == 0)
	evensum = evensum + i
	else
	oddsum = oddsum + i
		(13)	Understand	BTL2

13.	(i) State the rules for type checking with example. (ii) Give an algorithm for type inference and polymorphic function.	(7) (6)	Remember	BTL1
14.	Illustrate an algorithm for unification with its operation.	(13)	Apply	BTL3
15.	Write down the SDD for constructing syntax tree for the expression a+b*5	(13)	Understand	BTL2
16.	Illustrate in detail about Bottom-up evaluation of S-attribute definitions	(13)	Apply	BTL3

17.	Explain the evaluation order for SDD (13)	Evaluate	BTL5
	PART-C(15 -MARKS)		
1.	Create the following uind the arithmetic expression a+-(b+c)* into (i)Syntax tree (ii)Quadruples (iii)Triples (iv)Indirect Triples (15)	Create	BTL6
2.	Explain what is SDD and examine syntax-directed definition to differentiate expressions formed by applying the arithmetic operators + and * to the variable x and constants ; expression : x * (3 * x + x * x) (15)	Evaluate	BTL5
3.	Generate an intermediate code for the following code segment with the required syntax-directed translation scheme. (i) if (a > b) (7) x = a + b else x = a – b (ii) p>q AND r<s OR u>r (6)	Create	BTL6
4.	What is Type conversion? What are the two types of type conversion? Formulate the rules for the type conversion. (15)	Evaluate	BTL5
5.	Explain the specification of a simple Type Checkers (15)	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	Evaluate	BTL5

	computer.		
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
21	Examine what is the input to code generator.	Analyze	BTL4
22	What are the advantages and disadvantages of register allocation and assignments?	Understand	BTL2
23	How the use of registers is subdivided into 2 sub-problems?	Evaluate	BTL5
24	Organize the contents of activation record.	Apply	BTL3
	PART-B (13- MARKS)		
1.	(i)Illustrate the storage organization memory in the perspective of compiler writer with neat diagram. (8) (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 to reads nine integers into an array a and sorts them by using the concepts of activation tree. (9) (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	BTL2
6.	(i) Analyze the data access without nested procedure and the issues with nested procedure. (7) (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 (13)	Apply	BTL3

	example.		
13.	Discuss usage of stack in the memory allocation and discuss in detail about stack allocation space of memory. (13)	Understand	BTL2
14.	Describe the heap management of memory manager and locality of programs in detail . (13)	Remember	BTL1
15	Explain the problem that occurs in code generation with example (13)	Evaluate	BTL5
16	Illustrate the function of code generation algorithm in detail (13)	Analyze	BTL3
17	Discuss in detail about access links, manipulation of access links and access links for procedure (13)	Understand	BTL2
	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 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 tha t the smaller chunk can easily remain on the linked list of free space). However, we cannot tolerate chunks of fewer that 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. (15)	Evaluate	BTL5
2.	Explain the stack and heap allocation of memory in detail with suitable examples. (15)	Evaluate	BTL5
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	BTL6
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.	Create	BTL6
5	The following program is used to compute Fibonacci numbers recur•sively. Suppose that the activation record for f includes the following elements in order: (return value, argument n, local s, local t); there will normally be other elements in the activation record as well. The questions below assume that the initial call is f(5). (15) int f(int n) {	Evaluate	BTL5

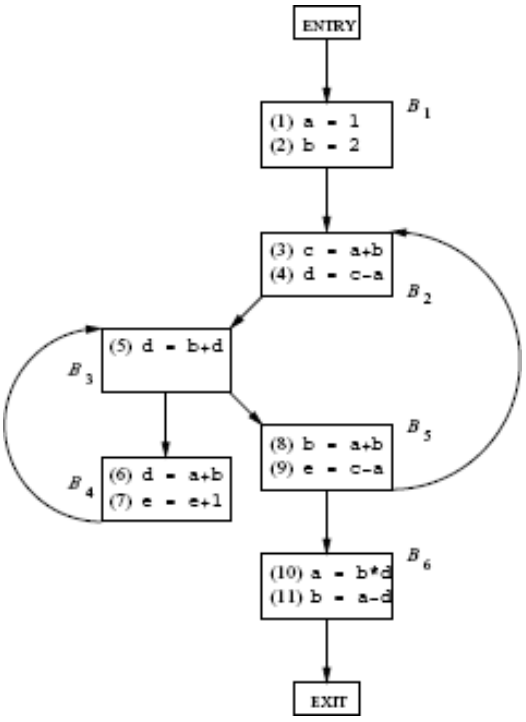
	int t, s;
	if (n < 2) return 1;
	s = f(n-1);
	t = f(n-2);
	return s+t;
	}
	
	a) Show the complete activation tree.
	b) What dose the stack and its activation records look like the first time f(1) is about to return?
	c) What does the stack and its activation records look like the fifth time f(1) is about to return?		

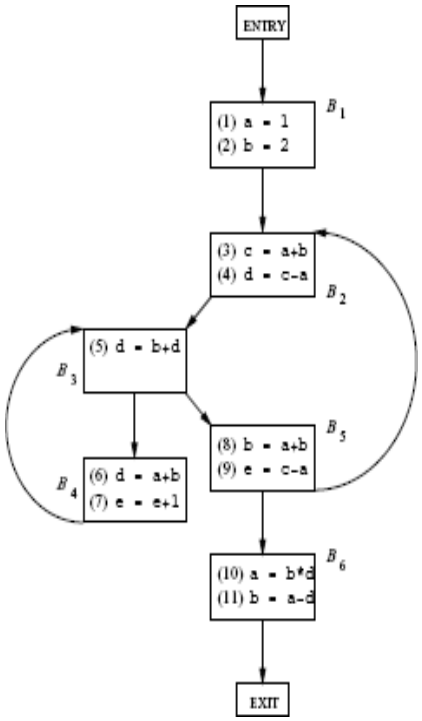
UNIT V- CODE OPTIMIZATION			
Principal Sources of Optimization – Peep-hole optimization - DAG- Optimization of Basic Blocks Global Data Flow Analysis - Efficient Data Flow Algorithm.			
PART-A (2 -MARKS)			
1.	List out the examples of function preserving transformations.	Remember	BTL1
2.	Illustrate the 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	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.	Analyze	BTL4
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 constant folding.	Understand	BTL2
14.	Prepare the three address code sequence for the assignment statement. d:= (a- b) + (a-c) + (a-c).	Create	BTL6
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

21	What is a flow graph?	Remember	BTL1
22	What is dead code elimination? Give example.	Understand	BTL2
23	Show an example for code motion.	Apply	BTL3
24	How the strength reduction is applied in code optimization?	Evaluate	BTL5
	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) t1: = 4 * i t2:= a [t1] t3: = 4 * i t4:= b [t3] t5:=t2*t4 t6:=Prod+t5 Prod:=t6 t7:=i+1 i:= t7 if i<= 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 the transfer equations for reaching definitions with example (13)	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 expressions. (8) (ii)Discuss in detail about the Use of Algebraic Identities. (5)	Understand	BTL2
10.	(i)Describe in detail about the flow of control optimization. (7) (ii)Identify the methods to eliminate the unreachable code, load and store data. (6)	Remember	BTL1
11.	(i)Give an example to identify the dead code in the DAG. (5) (ii)Describe the representation of array using DAG with example. (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. (7)	Create	BTL6

	(ii) Describe about induction variable and end reduction in strength	(6)		
14.	Describe the efficient data flow algorithms in detail.	(13)	Remember	BTL1
15	Explain in detail about optimization method performed on a small set of compiler generated instructions	(13)	Evaluate	BTL5
16	Discuss in detail about structure preserving transformation in detail	(13)	Understand	BTL2
17	Illustrate in detail about DAG Representation of basic block and Write algorithm for DAG Construction.	(13)	Apply	BTL3
	PART-C(15 MARKS)			

1.	Create DAG and three – address code for the following C program. (15)
	i = 1; s = 0;
	while (i<= 10)
	{
	s = s+ a[i] [i];
	i = i + 1;
	}
		(15)	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 Identify loop invariant computation for each loop	(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	(15)	Evaluate	BTL5
4.	Evaluate the available expressions on the following code by 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	(15)	Evaluate	BTL5
5.	Evaluate the Depth-first Ordering in iterative Algorithm and structure -Base Data flow Analysis in detail	(15)	Evaluate	BTL5