A1 and A2 slot: Last date: 30.09.2020

S.No	Topics
1)	1) Check the given Grammar is ambiguous/Unambiguous $S \rightarrow S(S)S \mid \varepsilon$ 2) Prove the given grammar is LR(1),LALR(1),Not SLR(1). $S \rightarrow Aa \mid bAc \mid dc \mid bda$ $A \rightarrow d$
2)	1) Find the following grammar is LL(1) ,LR(1) $S \rightarrow AaAb \mid BbBa$ 2) Check whether the following grammar is LR(0), SLR(1),LALR and LR(1) $S \rightarrow AaAb \mid BbBa$ $A \rightarrow \varepsilon$ $B \rightarrow \varepsilon$
3)	1) Construct DAG for a) (a-b)+c*(d/e) b) x=x+x*y c) (x+5)*(x+5+y) d) a=(a+a)+a(a+a+a)+a 2) Check the given grammer is LL(1) or NOT? $S \rightarrow (A) \mid 0$ $A \rightarrow SB$ $B \rightarrow ,SB \mid \varepsilon$ and also parse the grammar (0,(0,0))
4)	1. Explain& example following code optimization a) Common sub expression elimination b) Copy propagation c) Dead code elimination d) Code motion 2. Eliminate left recursion, perform left factoring and find: FIRST & FOLLOW $E \rightarrow E+T \mid T$ $T \rightarrow id \mid id\,[\,] \mid id\,[\,X\,]$ $X \rightarrow E,E \mid E$
5)	1) Consider the grammar ,compute the set of LR(0) items for the this grammar and draw the corresponding DFA $S \rightarrow S\ op\ S \mid x$ 2) What are the issuses of the lexical analyzer? 3) Explain the process of constructing an NFA from a regular expression.Find NFA for the expression $(a/b)^*$ a(a/b) (a/b) convert the obtained NFA into DFA
6)	1) Construct SLR parsing table $E \rightarrow E$ sub E sup E $E \rightarrow E$ sub E $E \rightarrow E$ sub E $E \rightarrow \{E\}$ $E \rightarrow c$ Resolve the conflict if any 2) What is an incremental compiler? Enlist the basic features of incremental compiler.

	3) With the help of a block schematic explain how “compiler-compiler” can reduce the effort in implementing new compiler.
7)	1) Find the predictive parser for the given grammar and parse the sentence (((a,a),↑,(a),a) S→a|↑|(T) T→T,S|S 2) Describe the principle sources of optimization
8)	1) Describe the function preserving transformation. 2) Construct: operator precedence relation and function for the given grammar E→E+E E→E*E E→id
9)	1) Explain the various issues in the design of code generation. 2) Explain code generation phase with simple code generation algorithm 3) Find Recursive desent parser E→iE’ E’→+iE’|ε
10)	1) What will be the precedence and associativity for the following grammar A→A$ B|B B→B#C/C C→C@D/D D→d 2) Construct DFA for the following regular expression $(a+b)^*$ (abb + a^+b) 3) Name different strategies that a perser can employ to recover from a syntactic error. Explain all of the strategies.
11)	1) Discuss the role of finite automata in compiler. 2) State what strategy LEX should adopt if keywords are not reserved words. 3) Write short note on input buffer with lexical analyser 4) Find FIRST and FOLLOW sets for the given grammer S→PQR P→a |Rb|ε Q→c|dp|ε R→e|f
12)	1) Discuss the role of finite automata in compiler. 2) State what strategy LEX should adopt if keywords are not reserved words. 3) Write short note on input buffer with lexical analyser 4) Find FIRST and FOLLOW sets for the given grammer S→PQR P→a |Rb|ε Q→c|dp|ε R→e|f
13)	1) With the help of neat block diagram explain various

	phases of compiler ,Also write down the output of each phase for expression a:=b+c*50 2) Construct LR(1). S→x|Ay B→ε|z A→Bx
14)	1) Consider the following CFG E→ TR R→+TR R→-TR R→ε T→num With the translation scheme to generate postfix expression equivalent to the given infix expression which is recognized by above grammar. All action in the translation should be at the the end of each production. 2) Find the predictive parser for the given grammar and parse the sentence (a+b)*c E→E+T|T T→T*F| F F→(E) |id
15)	1) Construct SLR parsing table for the grammar S→AB|gDa A→ab|c B→dC C→gC|g D→fD|g 2) Discuss the role of finite automata in compiler. 3) State what strategy LEX should adopt if keywords are not reserved words.
15)	1) Find the SLR parsing table for the given grammar and parse the sentence (a+b)*c E→E+E|E*E|(E)|id 2) Explain: Storage organization and Parameter passing
16)	1) Explain Model of activation record. 2) Construct LALR(1) parser table for the grammar S→iCtS|CtSeS|a C→b are there any conflicting entries?
17)	1) Construct CLR parsing table from S→AA A→Aa|b 2) Explain code motion, copy propagation ,dead code elimination.
18)	1) Give the criteria for achieving machine dependent optimization 2) Construct a canonical parsing table S→CC C→cC|d
19)	1) Give the criteria for achieving machine independent optimization. 2) Explain Global data flow analysis. 3) Construct the predictive parser table

A1 and A2 slot: Last date: 30.09.2020

<table>
<tr><td></td><td>S→(L)|a
L→L,S|S</td></tr>
<tr><td>20)</td><td>1) Explain Properties of optimizing compiler.
2) Construct SLR
E→E+T
T→T*F|F
F→F|0|1
3) Draw a transition diagram to recognize the C language operators ++,--,+=,-=,==,!=</td></tr>
<tr><td>21)</td><td>1) What do you mean by viable prefixes?
2) Find FIRST and FOLLOW
S→A
A→aBcB|B|D
B→dE
D→dE
E→FcA|FcC
C→c|ε
F→b
3) Translate the following assignment statement into Quadruples, Triples and Indirect Triples P=q*-r+q*-r</td></tr>
<tr><td>22)</td><td>1) How would you construct a DFA directly from a regular expression? Construct DFA for the regular expression (a/b) *abb#
2) Construct Predictive parsing
S→a|↑|(T)
T→ST’
T→,ST’|ε and also parsing (a,(a,a))</td></tr>
<tr><td>23)</td><td>1) Define handle and handle pruning?
2) Construct LR parsing table
E→E+T
E→T
T→T*F
T→F
F→(E)
F→id id*id+id using stack implementation.</td></tr>
<tr><td>24)</td><td>1) Differentiate between final states in a NFA and a DFA
2) Table:
<table>
<tr><th>Remove left recursion</th><th>Remove left Factoring</th></tr>
<tr><td>A→Aα |β</td><td>S→iEtS|iEtSeS|a
E→b</td></tr>
<tr><td>S→Aa|b
A→Ac|Sd|ε</td><td>Stmt→if expr then Stmt else Stmt|ifexpr then Stmt</td></tr>
<tr><td>S→aBDh
S→Bb|C
D→EF
E→g|ε
F→f|ε</td><td>S→aSb|aTc
T→dTU|ε
U→f</td></tr>
<tr><td>S→SA|SB|a|b|c</td><td></td></tr>
</table>
</td></tr>
<tr><td>25)</td><td>1) Construct LL (1)
S→iEtSS’|a</td></tr>
</table>

A1 and A2 slot: Last date: 30.09.2020

<table>
<tr><td></td><td>S’→eS|ε
E→b
2) Generate the three address code for
while(i<10)
{x=0;
i=i+1;}</td></tr>
<tr><td>26)</td><td>1) Explain Characteristics of Peephole optimization.
2) Consider the grammar
S→(L)|a
L→L,S|S
a) What are the terminal, non-terminal and start symbol?
b) Find parse tree for the following sentences
(i) (a,a)
(ii) (a,(a,a))
(iii) (a,((a,a),(a,a)))
c) Construct a LMD for each of the sentences in (b)
d) Construct a RMD for each of the sentences in (b)
e) What language does the grammar generate?</td></tr>
<tr><td>27)</td><td>1) Explain quadruple ,triple and three address code for (a+b)*(a+b)-(a+b)*d
2) Prove LR(0) =SLR(1)
S→bAc|Bc|d</td></tr>
<tr><td>28)</td><td>1) Explain briefly about compiler construction tools
2) Check the given grammar is LR(1) ,LALR(1) but not LL(0),LL(1)and LR(0)
S→X⍰
X→A1B
X→2
A→2
B→A</td></tr>
<tr><td>29)</td><td>1) Generate the three address code for the expression x:=A[i,j],for an array 10X20.Assume low_1 =1 and low_2=1
2) Check the given grammar is Not LL(1),NOT LR(0), LALR(1)
S→Xa⍰
X→Yb
X→YZc
Y→d
Z→b
3) Discuss the role of finite automata in compiler
4) What is L-attribute definition ? support your answer with example</td></tr>
<tr><td>30)</td><td>1) With the help of neat block diagram explain various phases of compiler ,Also write down the output of each phase for expression a:=b+c*50
2) Consider the following CFG
E→ TR</td></tr>
</table>

	R→+TR R→-TR R→ε T→num With the translation scheme to generate postfix expression equivalent to the given infix expression which is recognized by above grammar. All action in the translation should be at the the end of each production. 3) What is L-attribute definition? support your answer with example
31)	1) Construct LALR parsing table S→ L=R S→R S→*R L→id R→L 2) Construct DFA for the following regular expression $(a+b)^{*}$ (abb + a^{+}b)
32)	1) With the help of neat block diagram explain various phases of compiler ,Also write down the output of each phase for expression a:=b+c*50 2) Write a short note on data flow analysis. 3) Construct: operator precedence relation and function for the given grammar P→SR|S R→bSR|bS S→WbS|W W→L*W|L L→id
33)	1) Explain types of predictive parser. 2) Give the predictive parsing table for the following grammar E→E+T | T T→T*F|F F→(E) |id show the moves of the parser for the input(id+id*id)
34)	1) Explain Symbol table. 2) What are the properties of optimizing compiler. 3) Construct LR(1). S→x|Ay B→ε|z A→Bx
35)	1) Consider the grammar S→(L)|a L→L,S|S f) What are the terminal, non-terminal and start symbol? g) Find parse tree for the following sentences (iv) (a,a) (v) (a,(a,a))

	(vi) (a,((a,a),(a,a)))
36)	1) Define and compare:LR(0),LR(1), LALR(1),SLR(0),SLR(1),CLR(1) 2) Construct LALR(1) parser table for the grammar S→iCts|CtSeS|a C→b
37)	1) Define Activation tree 2) Explain about input buffering technique 3) Construct the NFA from the (a/b)*a(a/b) using Thompson's construction algorithm. 4) Find FIRST and FOLLOW sets for the given grammar S→PQR P→a |Rb|ε Q→c|dp|ε R→e|f
38)	1) What are the issues in the design of the code generator? Explain 2) The following grammar generates expressions formed by applying an arithmetic operator + to integer and real constants. When two integers are added, the resulting type is integer, otherwise it is real. E→E+T|T T→num.num|num Give a syntax directed translation scheme to determine the type of each subexpression. 3) Following grammar is for converting binary fraction to equivalent decimal value. [for example .101=0.625]. Give SDT scheme for the same. Also draw an annotated parse tree for .101 S→.L L→LB|B B→0|1
39)	1) What is the three address code? What are its types? How it is implemented. 2) What are the preliminary steps that are to be carried out during parsing?Explain the given grammar : S→a|↑|(T) T→T,S|S
40)	1) What are the issues of the Lexical analyser? 2) Eliminate left recursion, perform left factoring and find: FIRST & FOLLOW E→E+T|T T→id|id [ ] |id [X] X→E,E|E 3) Check the given grammar is LL(1) or NOT? S→(A) | 0 A→SB B→,SB|ε and also parse the grammar (0,(0,0))
41)	1) Explain the various data structurers used for implementing the symbol table and compare them. 2) The following grammar is not suitable for a top down

	predictive parser.Idetify the problem and correct it by rewriting the grammar. Show that your grammar satisfies LL(1). Condition by buldingLL(1) parse table for it. P→Ra|Qba R→aba|caba|Rbc Q→bbc|bc Also parse the string "cababca " using LL(1) table.
42)	1) Find: CLR (1) and LALR (1) (i) S→A A→AB|ε B→aB|b (ii) E→E+T T→T*F|F F→i 2) Check the given g is (option given) S→CC C →cC|d (1) LL(1) (2) SLR(1) but not LL(1) (3) LALR(1) but not SLR(1) (4) CLR(1) but not LALR(1) 3) S→AaAb|BbBa A→ε B→ε find LL(1),LR(0),SLR(1),CLR(1)and LALR(1). 5) Short Explain: Activation tree and principle sources of optimization
43)	1) S→Aa|bAc|dc|bda A→d find: find LL(1),LR(0),SLR(1),CLR(1)and LALR(1). 2) S→Aa|bAc|Bc|bBa A→d B→d find: find LL(1),LR(0),SLR(1),CLR(1)and LALR(1). 3) Short Explain: Activation tree and principle sources of optimization, input buffering technique.

A1 and A2 slot: Last date: 30.09.2020

Assignment Questions☹ 5 marks only

1) Find the given grammar :
S→CC
C →cC|d

LL(1)
SLR(1) but not LL(1)
LALR(1) but not SLR(1)
CLR(1) but not LALR(1)

2) Draw a transition diagram to recognize the C language operators ++,--,+=,-=,==,!=
3) Describe the principle sources of optimization
4) Find the predictive parser for the given grammar and parse the sentence (((a,a),↑,(a),a)
S→a|↑|(T)
T→T,S|S
5) Describe the principle sources of optimization

6) Consider the grammar
S→(L)|a
L→L,S|S
h) What are the terminal, non-terminal and start symbol?
i) Find parse tree for the following sentences (a,a) and (a,(a,a)) , (a,((a,a),(a,a)))
7) S→Aa|bAc|dc|bda
A→d find: find LL(1),LR(0),SLR(1),CLR(1)and LALR(1).
8) S→Aa|bAc|Bc|bBa
A→d
B→d find: find LL(1),LR(0),SLR(1),CLR(1)and LALR(1).
9) Short Explain: Activation tree and principle sources of optimization, input buffering technique

4) S→Aa|bAc|dc|bda
A→d find: find LL(1),LR(0),SLR(1),CLR(1)and LALR(1).
5) S→Aa|bAc|Bc|bBa
A→d
B→d find: find LL(1),LR(0),SLR(1),CLR(1)and LALR(1).
Short Explain: Activation tree and principle sources of optimization, input buffering technique.

1) How would you construct a DFA directly from a regular expression? Construct DFA for the regular expression (a/b) *abb#

A1 and A2 slot: Last date: 30.09.2020

2) Construct Predictive parsing
S→a|↑|(T)
T→ST'
T→,ST'|ε and also parsing (a,(a,a))

---

1) Describe the function preserving transformation.
2) Construct: operator precedence relation and function for the given grammar
E→E+E
E→E*E
E→id

---

1) Find the given grammar :
S→CC
C →cC|d

LL(1)
SLR(1) but not LL(1)
LALR(1) but not SLR(1)
CLR(1) but not LALR(1)

2) Draw a transition diagram to recognize the C language operators ++,--,+=,-=,==,!=
3) Describe the principle sources of optimization
4) Find the predictive parser for the given grammar and parse the sentence (((a,a),↑,(a),a)

S→a|↑|(T)
T→T¸S|S

---

1) Consider the grammar
S→(L)|a
L→L,S|S
j) What are the terminal, non-terminal and start symbol?
k) Find parse tree for the following sentences (a,a) and (a,(a,a)) , (a,((a,a),(a,a)))
2) S→Aa|bAc|dc|bda
A→d find: find LL(1),LR(0),SLR(1),CLR(1)and LALR(1).
3) S→Aa|bAc|Bc|bBa
A→d
B→d find: find LL(1),LR(0),SLR(1),CLR(1)and LALR(1).
Short Explain: Activation tree and principle sources of optimization, input buffering technique

---

1) peep hole optimization,
2) register allocation and assignment,
3) instruction selection by tree rewriting

4) S→Aa|bAc|dc|bda
A→d find: find LL(1),LR(0),SLR(1),CLR(1)and LALR(1).

A1 and A2 slot: Last date: 30.09.2020

5) S→Aa|bAc|Bc|bBa
A→d
B→d find: find LL(1),LR(0),SLR(1),CLR(1)and LALR(1).

6) Short Explain: Activation tree and principle sources of optimization, input buffering technique.