IMPORTANT THEORY QUESTION

MODULE-1

1. Define the following terms with example. (2M)
   a) String
   b) Alphabet
   c) Language
   d) DFA
   e) NFA
   f) ε-NFA
   g) ε-CLOSURE
   h) Powers of an Alphabet
   i) Equivalent states
   j) Distinguishable and indistinguishable states
2. Explain the differences between DFA and NFA

3. List and explain the various phases of compiler and show the output of each phase for the expression a=b + c * 25 (10M)

   or

   Explain the various phases of compiler and show the output of each phase for the expression position=initial + rate * 60
4. Explain the different phases of a compiler with an example. (10M)
5. With a neat diagram explain the language processing system. (06M)

MODULE -2

1. Define Regular Expression. (3M)
2. State and prove Pumping Lemma theorem. (6M)
3. Prove that for every RE, there is an equivalent finite automaton. (6M)
4. Explain input buffering strategy used in lexical analysis phase. (06M)
5. What is input buffering in lexical analysis? List the different methods of input buffering. Explain any one of them. (07M)
6. Define Tokens, Patterns and Lexemes. (03M)
7. Explain the role of lexical analyser in compiler. (5M)
8. Write regular definition for unsigned number and draw transition diagram for the same. (05M)
9. Construct transition diagram for recognizing relational operators. Sketch the program segment to implement it, showing the first state and one final state.
10. Construct the transition diagram to recognize the tokens of,
    a) a set of keywords like **begin**, **end**, **if**, **then** and **else**

b) identifiers
c) unsigned numbers
d) set of relational operators. (10M)

MODULE-3

1. Define Context free grammar.
2. Define the leftmost derivation, rightmost derivation and parse tree. (06M)
3. What is ambiguous grammar? Prove that the given grammar is ambiguous. (04M)
4. Give the rules for constructing FIRST and FOLLOW sets. (06M)
5. What is the role of parser? Explain the different error recovery strategies. (8M)

   OR

   With a schematic, explain the role of parser. List and explain various error recovery strategies of a parser. (10M)

6. Explain the left recursion and show how it is eliminated. Describe the algorithm used for eliminating the left recursion. (06M)

   OR

   How left recursion can be eliminated from grammars? Write down the simple arithmetic expression grammar and rewrite the grammar after removing left recursion.
7. Describe an algorithm used for eliminating the left recursion. Eliminate left recursion from the grammar:

   $S \rightarrow Aa \mid b \qquad A \rightarrow Ac \mid Sd \mid a.$

   (06M)
8. Explain the top down parsing and process for the string id+id*id. Given the grammar,
   $E \rightarrow E + E$
   $E \rightarrow E * E$
   $E \rightarrow (E)$
   $E \rightarrow id$
9. What are the key problems with top-down parse? Write a recursive descent parser for the grammar: $S \rightarrow cAd \quad A \rightarrow ab \mid a$

   OR

   Consider the production:
   $S \rightarrow aAb \quad A \rightarrow cd \mid c.$
   Show that recursive-descent parsing fails for the input string "acdb", also explain recursive descent algorithm.
10. Explain with a neat diagram, the model of a table-driven predictive parser.(8M)
11. Explain how error recovery is done in predictive parsing. [6 marks]

MODULE-4

1. Define Pushdown Automata. Explain the language of pushdown automata. (6M)
2. Define Handle, viable prefixes. (2M)
3. What is shift-reduce parser? Explain the conflicts that may occur during shift-reduce parsing. List the actions of shift-reduce parser.
4. What is handle pruning? Explain with example.
5. Write a schematic of LR parser.
6. Write an algorithm to construct SLR parsing table. (06M)
7. Write an algorithm used to compute LR (1) sets of items. (06M)
8. Write an algorithm for computation of CLOSURE of LR(0) items.
9. Write an algorithm used to compute LR (1) sets of items. (06M)

Module-5

1. Define a Turing machine. Explain the working of a Turing machine. (06M)
2. Explain extended TM(Multi-tape TM and Non-deterministic TM).(08M)
3. With a neat diagram, explain the working of basic Turing machine. (05M)
4. Prove that a multi tape TM is equivalent to a basic TM. (06M)
5. Prove that complement of a recursively enumerable language is recursive. (06M)
6. Write a note on universal TM and show that it simulates a computer. (08M)
7. Write short notes on (5M each):
   a) Multitape Turing machine
   b) Halting problem in TM
   c) Post correspondence problem
   d) Recursive languages
   e) Universal Turing machine
   f) Non-deterministic TM
   g) Decidable language
8. What is an attribute? Explain the different types of attributes with example. (08M)
9. What is Syntax Directed Definition? Write the grammar and SDD for a simple desk calculator and show annotated parse tree for the expression (3+4)*(5+6). (08M)
10. Write SDD for simple desk calculator suitable for top-down paring and show the annotated parse tree for the string **3*5**.
11. Give the Syntax Directed Definition to process a sample variable declaration

in C and construct dependency graph for the input **float x, y, z** (10M)

12. Write the grammar and syntax directed definition for a simple desk calculator and show annotated parse tree for the expression **3*5+4n**.
13. Explain synthesized attribute, inherited attribute, S-attributed definition and L-attributed definition with example for each. (10M)
14. Obtain the Directed Acyclic Graph for the expression**: a+a*(b-c)+(b-c)*d**. Also show the steps for constructing the DAG.
15. What is three address code? Explain different ways of representing three address codes with examples. (10M)
16. For the given expression **b* - c + b * - c** provide: i) TAC ii) Quadruple iii) Triple iv) Indirect Triple representation.
17. Translate the arithmetic expression: a+-(b+c) into quadruple, triple and indirect triples. (06M)
18. What is the difference between syntax tree and parse tree? (04M)
19. Explain the issues in the design of code generation. (10M)

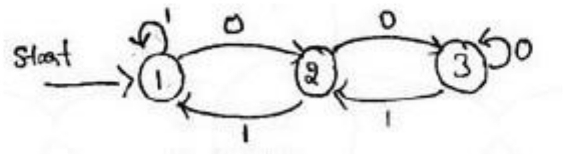
IMPORTANT PROBLEMS

Module-1

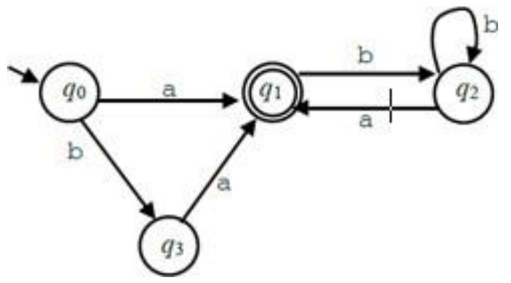
1. Construction of DFA and NFA: Refer class work problems. (2M or 3M each)
2. Conversion of NFA to DFA. (10M)
3. Minimize the given DFA. (10M)

Module-2

1. Construct Regular expressions for the given languages.(2M or 3M each)
2. Using $R_{ij}^{(k)}$/**Kleene's method** convert following FA to RE.



3. Using state elimination technique convert the following FA to RE.



4. Convert following RE's into FA.

5. (ab*)*
6. (a U b)*abb

7. Prove that following languages are not regular using Pumping Lemma.
   a. $L=\{ a^n b^n \mid n>=0\}$
   b. $L=\{ww^R \mid w\epsilon\{a,b\}^*\}$
   c. $L=\{a^P \mid P$ is a prime number$\}$
   d. $L=\{n_a(w)=n_b(w) \mid w\epsilon\{a,b\}^*\}$
   e. $L=\{a^{n!} : n>=0\}$

Module-3

1) Construct CFG for the following languages, (3marks each)
   i. $L = \{0^{2n} 1^m \mid m ,n \geq 0\}$
   ii. $L=\{a^n b^m c^n \mid m,n \geq 0\}$
   iii. $L = \{0^m 1^m 2^n \mid m\geq 1$ and $n\geq 0\}$
   iv. $L=\{0^i 1^j \mid i\neq j , i\geq 0, j\geq 0\}$
   v. $L=\{w: w\epsilon\{a, b\}^*$ and w is palindrome$\}$
   vi. $L=\{a^i b^j c^k : i+j=k, i>=0,j>=0\}$
   vii. $L=\{a^n b^m c^k : n+ 2m=k\}$
   viii. $L=\{0^i 1^j 2^k \mid i=j$ or $j=k\}$

2) Construct the predictive parsing table by making necessary changes to the grammar given below:
   $E \rightarrow E + T \mid T$
   $T \rightarrow T * F \mid F$
   $F \rightarrow (E) \mid id$
   (10M)
3) Consider the grammar
   $S \rightarrow aSbS \mid bSaS \mid \varepsilon$
   a. Show that this grammar is ambiguous by constructing two different leftmost derivations for the sentence **abab**.
   b. Construct the corresponding rightmost derivation for abab.
   c. Construct the corresponding parse trees for abab.
   d. What language does this grammar generate? (08M)
4) Show that the following grammar is ambiguous:
   a. $E \rightarrow E + E \mid E * E \mid (E) \mid id$.
   b. Write an equivalent unambiguous grammar for the same. (06M)
5) Given the grammar, $S\rightarrow(L) \mid a \quad L\rightarrow L, S \mid S$
   a. Make necessary changes to make it suitable for LL (1) parsing.
   b. Construction FIRST and FOLLOW sets
   c. Construct the predictive parsing table

d. Show the moves made by the predictive parser on input **(a, (a,a))** (12M)

6) Given the grammar:

   $S \rightarrow aABb$

   $A \rightarrow c \mid \varepsilon$

   $B \rightarrow d \mid \varepsilon$

   i)Compute FIRST and FOLLOW sets

   ii) Construct the predictive parsing table.

   iii) Show the moves made by predictive parser on the input; acdb

MODULE -4

1) Build a PDA to accept delimiters or balanced parenthesis having parenthesis {, (,), }. (08M)
2) Design a PDA for the language that accepts the string with $n_a(w)<n_b(w)$ where $w\epsilon(a+b)^*$ and show the instantaneous description of the PDA on input abbab. (10M)
3) Design a PDA to accept the language $L=\{a^n b^n : n>=1\}$(10M)
4) Design a PDA to accept the language $L=\{ a^n b^m a^n : n, m>=0\}$. (10M)
5) Design a PDA for the language that accepts the string with $n_a(w)=n_b(w)$ where,
6) $w \epsilon(a+b)^*$ and show the instantaneous description of the PDA on input abbaba. (10M)
7) Obtain a PDA for the language $L=\{wCw^R \mid w\epsilon(a+b)^*$ and w^R is the reverse of w by final state.(7M)
8) Consider the grammar $A\rightarrow(A) \mid a$. Construct the DFA of sets of LR (0) items. Show the parsing actions for input string **((a))**. Clearly show the states and symbols on the stack. (08M)
9) Construct LALR parsing table for the grammar given below using LR (1) items. (10M)

   $S\rightarrow CC$

   $C\rightarrow cC \mid d$

10) Given the grammar:

   $S\rightarrow CC$

   $C\rightarrow cC \mid d$

   a) Obtain the sets of canonical collection of sets of valid LR (0) items.
   b) Design SLR parsing table. (10M)

11) a) Obtain LR (0) items for the following grammar. (08M)

   $S\rightarrow L=R \mid R$

   $L\rightarrow * R\mid id$

   $R\rightarrow L$

   b)Obtain FIRST and FOLLOW sets for the grammar shown in (a) and obtain SLR parsing table. Is the grammar SLR? (12M)

12) Write the canonical collection of LR(0) items and SLR parsing table for the following grammar: (14M)

$E \rightarrow E + T \mid T$
$T \rightarrow T*F \mid F$
$F \rightarrow (E) \mid id$

MODULE -5

1) Design Turing machine to accept language, L= $\{0^n1^n: n\geq1\}$. Also show sequence of moves made by the TM for the string "000111". (12M)
2) Design Turing machine to accept language, L= $\{ww^R: w\in\{a,b\}^*\}$. Write its transition diagram. Also show sequence of moves made by the TM for the string "aabbaa". (14M)
3) Design a Turing machine to accept L= $\{0^n1^n2^n \mid n>=0\}$. Draw the transition diagram. Show the moves made for string 001122. (12M)
4) Construct a Turing machine that can accept the set of all even palindromes over {0,1}.
5) Design a Turing machine to accept L={ w$\in${a, b} and $n_a(w)=n_b(w)$ } (08M)

Note:

- DFA can also be called DFSM, similarly NFA as NDFSM.
- (a+b) can be written as (aUb) also.
- Non recursive predictive parsing or LL(1) parser are same.
- CLR(1) can also be written as LR(1)
- All the problems refer ATC class work.
- All the above-mentioned questions are frequently asked in exams. But don't expect same questions in the exam.
- Prepare for both ATC and compiler part as it can be merged in both 1st part and 2nd part of each questions.