Assignment 1

1. What is a symbol table? Discuss the most suitable data structure for it by stating merits / demerits.
2. Explain different phases of compiler
3. Define lexeme, token and pattern. Identify the lexemes that make up the tokens in the following program segment. Indicate corresponding token and pattern.

   void swap (int a, int b)

   {

   int k;

   k = a;

   a = b;

   b = k;

   }
4. Construct DFA for following Regular expression. Use firstpos, lastpos and followpos functions to construct DFA.

   (a * | b *) *
5. What is regular expression, give all the algebraic properties of regular expression
6. Construct DFA without constructing NFA for following regular expression:

   a*b*a(a | b)b*a#

   Minimize the same
7. Construct NFA for following regular expression using Thompson's notation and then convert it into DFA.

   a(b | c)*a*c#
8. Construct NFA for following Regular Expression using Thomson's Construction.

   Apply subset construction method to convert into DFA.

   (a | b)*abb
9. Construct DFA by syntax tree construction method.

   a+ b* (c |d) f #

   Optimize the resultant DFA.

10. Explain linker & loader.
11. Compare top-down and bottom-up parser.
12. Explain right-most-derivation-in-reverse with the help of an example.
13. Explain SLR parser. How is its parse table constructed?
14. Explain all error recovery strategies using suitable examples
15. Consider the grammar

    S -> SS+ | SS* | a

    Show that the string aa+a* can be generated by the grammar. Construct the parse tree for the grammar. Is the grammar ambiguous?

    Write unambiguous production rules for if then else construct.
16. Apply shift reduce parser for parsing following string using unambiguous grammar.

    id - id * id - id
17. Where do we use operator precedence parsing technique? Give the general precedence table for operating precedence parsing, considering all the generalized rules.
18. Construct a precedence graph, precedence table for operator precedence parser to be used for parsing a string consisting of id, - , * , $. Parse following string.

    $ id - id * id * id $
19. What is left recursion? Eliminate the left recursion from the following grammar. And

    Design the FIRST SET and FOLLOW SET for the following grammar

    E ◊ E + T | T

    T ◊ T * F | F

    F ◊ (E) | id
20. Explain left factoring with the help of an example.
21. Differentiate SLR, Canonical LR and LALR. Also justify the statement “A class of grammar that can be parsed using LR methods is a proper subset of the class of grammars that can be parsed with predictive parser”
22. Check whether the following grammar is CLR or not.

    S ◊ Aa | bAc | Bc | bBa

    A ◊ d

    B ◊ d

23. For the following grammar

    D ◊ T L ;

    L ◊ L , id | id

    T ◊ int | float

    1)Remove left recursion (if required)

    2)Find first and follow for each non terminal for Resultant grammar

    3)Construct LL(1) parsing table

    4)Parse the following string (show stack actions clearly) and

24. Unsigned numbers are strings such as 5280, 39.37, 6.336E4 or 1.894E-4, give the regular definitions for the above mentioned strings.

25. Draw the state transition diagram for the unsigned numbers

Subject Teacher **H.O.D**

Mr. Krunal V. Patel **Mr. Krunal V. Patel**