|  |
| --- |
| 1.a.Describe **the** Role of the lexical analyzer generator  b. Discuss Recognition of tokens.  2.Explain optimization of DFA-based pattern matchers |
|
| 3.Construct an SLR parsing table for the following grammar:  E -> E + T/T T -> T \* F/F and F -> (E)/id.   1. Explain the stack implementation of the shift-reduce parsing method with an example. Design shift-reduce parser for the grammar E->E\*E|E+E|E-E|(E)|id and the input string w=id+id\*id. |
|
| 4.Construct optimized DFA for the R.E (a|/b)\*abb  5.Demonstrate phases of compilers in detail with an example  c=a\*b+c\*d  Construct LL(1) parsing table for the given grammar  S -> iEtS| iEtSeS’| a , S’->eS/€ , and E->b  6.(a) What are the limitations of a recursive descent parser?  (b) What is left recursion? Remove left recursion from following grammar:  S -> Aa/b A -> Ac/Sd  7.Draw the block diagram of the LR parser. Compute LR(0) items for the grammar E -> E + T/T T -> T \* F/F and F -> (E)/id.  8.Construct the CLR canonical forms S->CC  C->aC/d .  9. Construct LALR Parsing table for the grammar  S→L=R/R  L→\*R/id  R→L  10. Construct predictive parsing table for the following grammar  S → (L) | a  L → L, S | S  SHORT QUESTIONS: |
|
|
|
|
|
|
|

|  |
| --- |
| Define lexme,pattern,token |
| List out the rules to calculate First and Follow functions. |
| E -> TE’ , E’ -> +T E’|Є , T -> F T’ , T’ -> \*F T’ | Є , F -> (E) | id  Calculate first set the for the above grammar |
| Define Left recursion. |
| Define Finite automata |

|  |
| --- |
| List out various types of language translators |
| Write first and last position for (a/b)\*.a ? |
| List and explain various error recovery strategies. |
| What is bottom-up parsing? List various bottom up parsers |
| Distinguish left recursion and left factoring. |