|  |
| --- |
| **Name:**Liya Elizabeth Joe  **RollNo:**48 |
| **Exp. No:**14 **Date:**17/11/2020 |
| **Study of Yacc** |
| **Aim:** To familiarize with basic concepts of Yacc. |
| YACC (yet another compiler-compiler) is anLALR(1) (LookAhead, Left-to-right, Rightmost derivation producer with 1 lookahead token) parser generator that takes as input a specification of a syntax, and produces as output a procedure for recognizing that language.YACC was designed for being complemented by Lex.  **Structure of Yacc**  Input to YACC is divided into 3 sections.  Definitions  %%  Rules  %%  Subroutines   * **Definitions** consist of token declarations and C code bracketed by %{ %}. * **Rules**: BNF grammar is placed in the rules section.(Backus Naur Form).   BNF grammars are used to express CFG.L.H.S of production or non terminal is entered left justified followed by a colon.This is followed by the RHS of the production .Actions associated with rules are enclosed in braces.The RHS of production is replaced in the parse stack with LHS of the same production.The positions in the value stack is referenced in the C code by specifying $.   * **Subroutines**: Consist of C statements.   **Built-in Functions in Yacc**  Some of the built in functions used in Yacc are:   1. **yyerror()** : It’s a user specified error function.In the event of syntax errors YACC calls yyerror. 2. **yyparse()** : It returns a value 0 if the input it parses is valid according to the   given grammar rules.It returns 1 if parsing failed because of invalid input i.e. the input contains syntax errors.It returns 2 if parsing is failed due to memory exhaustion.   1. **yylval()** : The values associated with tokens are returned by lex in variable yylval. 2. **yystype()** : The type of yylval is determined by yystype.   **Compilation and Execution**  The steps for compilation and execution of a Yacc program are as follows:   1. Compile lex file:lex filename.l 2. Compile yacc file:yacc -d filename.y 3. gcc lex.yy.c y.tab.c 4. ./a.out   **Sample Program**  Lex program  %{  #include<stdio.h>  #include "y.tab.h"  extern int yylval;  %}  %%  [0-9]+ {  yylval=atoi(yytext);  return NUMBER;    }  [\t] ;    [\n] return 0;    . return yytext[0];    %%    int yywrap()  {  return 1;  }  Yacc program  %{  #include<stdio.h>  int flag=0;  %}  %token NUMBER  %left '+' '-'  %left '\*' '/' '%'  %left '(' ')'  %%  ArithmeticExpression: E{    return 0;  };  E:E'+'E {$$=$1+$3;}  |E'-'E {$$=$1-$3;}  |E'\*'E {$$=$1\*$3;}  |E'/'E {$$=$1/$3;}  |E'%'E {$$=$1%$3;}  |'('E')' {$$=$2;}  | NUMBER {$$=$1;}  ;  %%  void main()  {  printf("\nEnter the expression \n");  yyparse();  if(flag==0)  printf("\nEntered arithmetic expression is Valid\n\n");  }  void yyerror()  {  printf("\nEntered arithmetic expression is Invalid\n\n");  flag=1;  } |
| **Result:** Familiarized with basic concepts of Yacc. |
| **Remarks:**(To be filled by faculty) |
| **Sample Input and Output** |