**Ex. No. 4) A IMPLEMENTATION OF PARSER USING YACC**

**Date:**

**Aim:**

To Generate YACC specification to recognize a valid arithmetic expression that uses operator +,- ,\* and /.

**Algorithm:**

1. Start the program.
2. In lex file, specify RE in order to recognize tokens and return tokens to syntax analysis.
3. In yacc file, specify CFG and their actions are to be done on tokens that are from lexical analyzer.
4. In main(), yyparse() is used to initialize the syntax analysis.
5. yyerror() is used to throw an error when it is encountered.
6. Input the arithmetic expression and recognize it whether it is valid or not.
7. Stop the program.

**Program:**

**lextoollexfile.l:**

%{

#include<stdio.h>

#include"y.tab.h"

extern yylval;

%}

%%

[0-9]+ {yylval=atoi(yytext);return NUMBER;}

[+] {return '+';}

[\*] {return '\*';}

[-] {return '-';}

[/] {return '/';}

[\t]+;

[\n] {return 0;}

. {return yytext[0];}

%%

int yywrap()

{

return 1;

}

**yaccfile.y**

%{

#include<stdio.h>

%}

%token NUMBER

%left '+'

%right '\*'

%left '-'

%left '/'

%%

start: E {}

E: E '+' E {$$=$1+$3;}

| E '\*' E {$$=$1\*$3;}

| E '-' E {$$=$1-$3;}

| E '/' E {$$=$1/$3;}

|'(' E ')' {$$=$2;}

| NUMBER {$$=$1;}

;

%%

void main(){

printf("\n Enter the Arithmetic Expression:\n");

if(yyparse()==0)

printf("\nValid Expression.\n");

}

yyerror(){

printf("Invalid Expr.\n");}

**Output:**

root@cselab-OptiPlex-380:~/Desktop# lex lextoollexfile.l

root@cselab-OptiPlex-380:~/Desktop# yacc -d yaccfile.y

root@cselab-OptiPlex-380:~/Desktop# gcc lex.yy.c y.tab.c

root@cselab-OptiPlex-380:~/Desktop# ./a.out

Enter the Arithmetic Expression:

(2/3)-(2+4)/(3\*1)

Valid Expression.

**Result:**

Thus, the yacc specification to recognize valid arithmetic expression was implemented and executed successfully.

**Ex. No. 4) B IMPLEMENTATION OF PARSER USING YACC**

**Date:**

**Aim:**

To generate YACC specification to recognize valid variables.

**Algorithm:**

1. Start the program.
2. In lex file, specify RE in order to recognize tokens and return tokens to syntax analysis.
3. In yacc file, specify CFG and their actions are to be done on tokens that are from lexical analyzer.
4. In main(), yyparse() is used to initialize the syntax analysis.
5. yyerror() is used to throw an error when it is encountered.
6. Input the variables and recognize it whether it is valid or not.
7. Stop the program.

**Program:**

**ExNo4b.l:**

%{

#include<stdio.h>

#include"y.tab.h"

extern yylval;

%}

%%

[\_]?[a-zA-Z\_][a-zA-z0-9\_]\* {return IDENTIFIER;}

[\t]+;

[\n] {return 0;}

. {return yytext[0];}

%%

int yywrap()

{

return 1;

}

**ExNo4b.y**

%{

#include<stdio.h>

%}

%token IDENTIFIER

%%

start: E {}

E: IDENTIFIER {$$=$1;}

;

%%

void main()

{

printf("\nEnter identifier :\n");

if(yyparse()==0)

printf("valid identifier \n");

}

yyerror()

{

printf("Invalid identifier\n");

}

**Output:**

welcome@welcome-HP-Notebook:~/Desktop$ lex ExNo4b.l

welcome@welcome-HP-Notebook:~/Desktop$ yacc -d ExNo4b.y

welcome@welcome-HP-Notebook:~/Desktop$ gcc lex.yy.c y.tab.c

welcome@welcome-HP-Notebook:~/Desktop$ ./a.out

Enter identifier :

\_123

valid identifier

welcome@welcome-HP-Notebook:~/Desktop$ ./a.out

Enter identifier :

fdbdf123\_

valid identifier

welcome@welcome-HP-Notebook:~/Desktop$ ./a.out

Enter identifier :

123\_gff

Invalid identifier

welcome@welcome-HP-Notebook:~/Desktop$ ./a.out

Enter identifier :

\_

valid identifier

welcome@welcome-HP-Notebook:~/Desktop$ ./a.out

Enter identifier :

\_\_\_

valid identifier

**Result:**

Thus, the yacc specification to recognize valid varaiables was implemented and executed successfully.

**Ex. No. 4) C IMPLEMENTATION OF PARSER USING YACC**

**Date:**

**Aim:**

To Generate YACC specification to implement a calculator.

**Algorithm:**

1. Start the program.
2. In lex file, specify RE in order to recognize tokens and return tokens to syntax analysis.
3. In yacc file, specify CFG and their actions are to be done on tokens that are from lexical analyzer.
4. In main(), yyparse() is used to initialize the syntax analysis.
5. yyerror() is used to throw an error when it is encountered.
6. Input the expression of values and display the result of the expression.
7. Stop the program.

**Program:**

**lextoollexfile1.l:**

%{

#include<stdio.h>

#include"y.tab.h"

extern yylval;

%}

%%

[0-9]+ {yylval=atoi(yytext);return NUMBER;}

[+] {return '+';}

[\*] {return '\*';}

[-] {return '-';}

[/] {return '/';}

[\t]+;

[\n] {return 0;}

. {return yytext[0];}

%%

int yywrap()

{

return 1;

}

**yaccfile1.y**

%{

#include<stdio.h>

%}

%token NUMBER

%left '+'

%right '\*'

%left '-'

%left '/'

%%

start: E {printf("%d",$$);}

E: E '+' E {$$=$1+$3;}

| E '\*' E {$$=$1\*$3;}

| E '-' E {$$=$1-$3;}

| E '/' E {$$=$1/$3;}

|'(' E ')' {$$=$2;}

| NUMBER {$$=$1;}

;

%%

void main(){

printf("\n Enter the Arithmetic Expression:\n");

if(yyparse()==0)

printf("\nValid Expression.\n");

}

yyerror(){

printf("Invalid Expr.\n");}

**Output:**

root@cselab-OptiPlex-380:~/Desktop# lex lextoollexfile1.l

root@cselab-OptiPlex-380:~/Desktop# yacc -d yaccfile1.y

root@cselab-OptiPlex-380:~/Desktop# gcc lex.yy.c y.tab.c

root@cselab-OptiPlex-380:~/Desktop# ./a.out

Enter the Arithmetic Expression:

(2\*6)-(3/2)+(5\*5)

36

Valid Expression.

**Result:**

Thus, the yacc specification for calculator was implemented, executed and output was verified successfully.