Expt. No. 2

Page No. 4

Expt. Name. Use LEX and YACC to implement parser on ambigous grammar.

Date :

Aim:

To write the program using LEX and YACC to implement parser on ambigous grammar

~~B~~ Algorithm: → File.l

```
step 1: Start
step 2: Include the necessary header files and declare the
        necessary variables.
step 3: initialize the digits, operators, parenthesis and return the
        value else print ~~[illegible]~~ Syntax Error
step 4: Call the function & return 1
step 5: Stop
```

File.y

```
step 1: Start
step 2: Include the necessary header files and declare the
        necessary variables
step 3: Substitute the values and Calculate respectively for
        Addition, Subtraction, Multiplication and division and
        return the result.
step 4: Call the main function and print the result
step 5: Stop.
```

Expt. No.

Expt. Name.

Date :

Program:

File.l

```
% option noyywrap
% {
        #include<stdio.h>
        #include "y.tab.h"
        void yyerror (char *s)
        extern int yylval;
% }
%%
[0-9]+ {yylval=atoi(yytext);
return NUM;
[-+*/\n]    {return *yytext;}
"("          {return *yytext;}
")"          {return *yytext;}
[\t];
.            {yyerror("Syntax Error");}
%%
int yywrap()
 {
 return 1;
 }
```

File.y

```
% {
        #include<stdio.h>
        extern Int yylex(void);
        void yyerror(char *);
% }
```

Expt. No.

Expt. Name.

Date :

```
%token NUM
%%
S:
S expr '\n'  {printf("%d\n",$2);}
|
;
expr:
 expr '+' expr      {$$ = $1 + $3;}
|expr '-' expr      {$$ = $1 - $3;}
|expr '*' expr      {$$ = $1 * $3;}
|expr '/' expr      {$$ = $1 / $3;}
| NUM               {$$ =$1;}
| '('expr')'        {$$ =$2;}
;
%%
void yyerror(char *s)
{
printf ("%s\n",s);
}

int main()
{
yyparse();
return 0; }
```

<u>Result:</u>

Use LEX and YACC to implement parser for ambiguous is executed successfully.