```
%{
#include "y.tab.h"
#include<math.h>
%}

NUMBER [0-9]+|([0-9]*" . "[0-9]+)

%%
{NUMBER}  {yylval.dval=atof(yytext);
                                        return NUMBER;}

[\t]+;
\n|. return yytext[0];
%%



%{
#include "y.tab.h"
#include<stdio.h>
#include<math.h>
%}

%{
int yylex();
void yyerror(const char *s);
%}

%union
```

```
{

double dval;

}

%token <dval> NUMBER
%left '+' '-'
%left '*' '/'
%type <dval> E

%%

SL : S '\n';
S : E {printf("=%f\n",$1);};
E : E '+' E{$$=$1+$3;} | E '-' E{$$=$1-$3;} | E '*' E{$$=$1*$3;} | E '/' E{if($3==0)

                                 printf("Error Divide By Zero");

                                 else

                                 $$=$1/$3;

                                  printf("Division is");} | NUMBER {$$=$1;};
%%
extern FILE*yyin;

int main()
{
```

```
do
{

yyparse();

}while(!feof(yyin));

}

yyerror(char*a)
{

fprintf(stderr,"parse error");

}
/*
stud@stud-OptiPlex-380:~/Desktop$ lex ass6.l
stud@stud-OptiPlex-380:~/Desktop$ yacc -d cal.y
stud@stud-OptiPlex-380:~/Desktop$ gcc y.tab.c lex.yy.c -ll -lm
stud@stud-OptiPlex-380:~/Desktop$ ./a.out
2*3
=6.000000
*/
```