ao.l

```
%{
#include<stdio.h>
#include"aro.tab.h"
extern int yylval;
%}
%%
[0-9]+ {
        yylval=atoi(yytext);
        return NUM;
      }
[\t] ;
\n return 0;
. return yytext[0];
%%
```

aro.y

```
%{
#include<stdio.h>
#include<conio.h>
%}
%token NUM
%left '+' '-'
%left '*' '/'
%left '(' ')'
%%
expr: e{
       printf("result:%d\n",$$);
       return 0;
      }
e:e'+'e {$$=$1+$3;}
 |e'-'e {$$=$1-$3;}
 |e'*'e {$$=$1*$3;}
 |e'/'e {$$=$1/$3;}
 |'('e')' {$$=$2;}
 | NUM {$$=$1;}
;
%%
```

```
int main()
{
   printf("\n enter the arithematic expression:\n");
   yyparse();
   printf("\nvalid expression\n");
}
yyerror()
{
   printf("\n invalid expression\n");
 }
int yywrap()
{
return 1;
}
```

```
flex ao.l
bison -d aro.y
gcc lex.yy.c aro.tab.c
```