

1.Cal.l

```
%{
#include <stdio.h>
void yyerror(char *s);
#include "y.tab.h"

}%

%%
[0-9]+ { yylval = atoi(yytext);return INTEGER; }
[() + * \n] return *yytext;
[ \t] ; /* skip whitespace */
. yyerror("invalid character");
%%
int yywrap(void){
return 1;
}
```

1.cal.y

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

%token INTEGER
%left '+'
%left '*'

%%
lines : lines expr '\n' { printf("%d\n", $2); }
      | lines '\n' {}
      ;

expr : expr '+' expr { $$ = $1 + $3; }
     | expr '*' expr { $$ = $1 * $3; }
     | '(' expr ')' { $$ = $2; }
     | INTEGER { $$ = $1; }
     ;

%%

int main(void) {
yyparse();
return 0;
}
void yyerror(char * s) {
fprintf(stderr, "%s\n", s);
}
```

```
}
```

```
2.calaa.l
```

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

```
%token INTEGER
```

```
%left '+'
```

```
%left '*'
```

```
%%
```

```
E1 : E '\n' { printf("valid"); }
```

```
    |
```

```
    ;
```

```
E : E '+' E { }
```

```
    | E '*' E { }
```

```
    | '(' E ')' { }
```

```
    | INTEGER { }
```

```
    ;
```

```
%%
```

```
int main(void) {
```

```
    yyparse();
```

```
    return 0;
```

```
}
```

```
void yyerror(char * s) {
```

```
    fprintf(stderr, "%s\n", s);
```

```
}
```

```
3.deskcal.l
```

```
%{
```

```
#include <stdio.h>
```

```
#include "y.tab.h"
```

```
extern int yylval;
```

```
%}
```

```
%%
```

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

```
[() + * \n] {return (*yytext);}
```

```
[\t] { } /* skip whitespace */
```

```
%%
```

```
int yywrap(void){
```

```
    return 1;
```

```

}
3.deskcalnew.l

%{
/* Definition section */
#include<stdio.h>
#include "y.tab.h"
extern int yylval;
}%

/* Rule Section */
%%
[0-9]+ {
    yylval=atoi(yytext);
    return NUMBER;

}
[\t] ;

[\n] return 0;

. return yytext[0];

%%

int yywrap()
{
return 1;
}

3.deskcalnew.y
%{
/* Definition section */
#include<stdio.h>
int flag=0;
}%

%token NUMBER

%left '+' '-'

%left '*' '/' '%'

%left '(' ')'

/* Rule Section */
%%

ArithmeticExpression: E{

    printf("\nResult=%d\n", $$);

```

```

        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;}

;

%%

//driver code
void main()
{
printf("\nEnter Any Arithmetic Expression which can have operations Addition,Subtraction,
Multiplication, Division,Modulus and Round brackets:\n");

yyparse();
if(flag==0)
printf("\nEnter arithmetic expression is Valid\n\n");
}

void yyerror()
{
printf("\nEnter arithmetic expression is Invalid\n\n");
flag=1;
}

4.first.l

%option noyywrap
%{
#include<stdio.h>
extern int yylval;
int NUMBER;
}%
%%
[0-9]+ {yylval=atoi(yytext);return(NUMBER);}
. {return yytext[0];}
%%

```

4.first.y

```

%{
#include<stdio.h>
int yylex();
}%

%token NUMBER
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'
%%
A : '\n' E '\n' { printf("\nResult=%d\n", $2); }
| {}
;
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("ENter Arithmetic Expression");
yyparse();
}
void yyerror(){
printf("\nENtered AE is invaid");
}

```

6.firsty.l

```

%option noyywrap
%{
#include<stdio.h>
#include "y.tab.h"
extern int yylval;
}%
%%
[0-9]+ {yylval=atoi(yytext);return(NUMBER);}
[+*] {return(yytext[0]);}
. {return yytext[0];}
%%

```

firsty.y

```

%{
#include<stdio.h>
int yylex();
}%

```

```

%token NUMBER

```

```

%left '+' '-'
%left '*' '/' '%'
%left '(' ')'
%%
A : E '\n' { printf("\nResult=%d\n", $2); }
|
;
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("ENter Arithmetic Expression");
yyparse();
}
void yyerror(){
printf("\nENtered AE is invaid");
}

```

7.in_to_post.l

```

%{
/* Definition section */
%}
ALPHA [A-Z a-z]
DIGIT [0-9]

/* Rule Section */
%%
{ALPHA}({ALPHA}|{DIGIT})* return ID;
{DIGIT}+ {yylval=atoi(yytext); return ID;}
[\n \t] yyterminate();
. return yytext[0];
%%

```

7.in_to_post.y

```

%{
/* Definition section */
#include <stdio.h>
#include <stdlib.h>
%}

%token ID
%left '+' '-'
%left '*' '/'
%left UMINUS

```

```

/* Rule Section */
%%

S : E
E : E'+'{A1();}T{A2();}
  | E'-'{A1();}T{A2();}
  | T
  ;
T : T'*'{A1();}F{A2();}
  | T/'{A1();}F{A2();}
  | F
  ;
F : '(E{A2();})'
  | '-'{A1();}F{A2();}
  | ID{A3();}
  ;

%%

#include "lex.yy.c"
char st[100];
int top=0;

//driver code
int main()
{
    printf("Enter infix expression: ");
    yyparse();
    printf("\n");
    return 0;
}

A1()
{
    st[top++]=yytext[0];
}

A2()
{
    printf("%c", st[--top]);
}

A3()
{
    printf("%c", yytext[0]);
}

8.paranthesis_count.l
%{
#include "y.tab.h"
extern int yylval;
%}

```

%%

[\\n] yyterminate();

%%

```
int yywrap(void){
return 1;
}
```

8.paranthesis_count.y

%{

```
#include <stdio.h>
#include <stdlib.h>
void yyerror(char *s);
int c=0;
%}
```

%token '(' ')'

%%

```
E : S {printf("%d\\n",c);}
S : '(' S ')' {c+=1;}
S : ;
```

%%

```
int main()
{
    printf("Enter expression: ");
    yyparse();
    printf("%d\\n",c);
    return 0;
}
void yyerror(char * s) {
    fprintf(stderr, "%s\\n", s);
}
```

9.pos_neg.l

%option noyywrap

%{

```
#include<stdio.h>
```

```
int neg=0,pos=0;
```

%}

%%

```
[-][0-9]+ {neg++;}
```



```

[0-9]+ {pos++;}
[\n] {return 0;}
%%

int main(){
yylex();
printf("Negative count is %d",neg);
printf("Positive count is %d",pos);
return 0;
}

10.post_eval.l
%{
#include "y.tab.h"
extern int yylval;
%}

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

[-+*/] {return yytext[0];}
[\n] yyterminate();

%%

int yywrap(void){
return 1;
}

10.post_eval.y
%{
#include<stdio.h>
#include<assert.h>
void push(int val);
int pop(void);
void yyerror(char *s);

%}

%token ID

%%

S : E {printf("= %d\n",top());}
;
E : E E '+' {push(pop()+pop());}
| E E '-' {int temp=pop();push(pop()-temp);}
| E E '*' {push(pop()*pop());}
| E E '/' {int temp=pop();push(pop()/temp);}
| ID {push(yylval);}
;

```

%%

```
int st[100];  
int i=0;
```

```
void push(int val)  
{  
    assert(i<100);  
    st[i++]=val;  
  
}
```

```
int pop()  
{  
    assert(i>0);  
    return st[--i];  
  
}
```

```
int top()  
{  
    assert(i>0);  
    return st[i-1];  
}
```

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

```
void yyerror(char * s) {  
    fprintf(stderr, "%s\n", s);  
}
```