1. a) Write a LEX program to recognize valid *arithmetic expression.* Identifiers in the expression could be only integers and operators could be + and *. Count the identifiers & operators present and print them separately.

<u>1a.l</u>

```
%{
#include<stdio.h>
int b=0,op=0,id=0;
%}
%%
[A-Za-z][A-Za-z0-9]*          {id++;printf("\nIdentifier:");ECHO;}
[\+\-\*\/]                    {op++;printf("\nOperator:");ECHO;}
"("                           {b++;}
")"                           {b--;}
.|\n                          {;}
%%
main()

{
        printf("Enter the expression\n");
        yylex();
        printf("\nTotal no. of identifiers%d",id);
        printf("\nTotal no. of operators%d",op);
        if((op+1)==id&&v==0)
                printf("\nExpression is valid\n");
        else
                printf("\nExpression is invalid\n");
}
```

<u>Execution Steps:</u>

```
lex 1a.l
cc lex.yy.c –ll
. /a.out
```

1. b)Write YACC program to evaluate *arithmetic expression* involving operators: +, -, *, and /

<u>1b.l</u>

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

<u>1b.y</u>

```
%{
#include<stdio.h>
#include<stdlib.h>
%}
%token NUM
%left '+' '-'
%left '*' '/'
%%
input:exp {printf("%d\n",$1);exit(0);}
exp:exp'+'exp {$$=$1+$3;}
|exp'-'exp {$$=$1-$3;}
|exp'*'exp {$$=$1*$3;}
|exp'/'exp {if($3==0){printf("Divide by zero error\n");exit(0);}
else
$$=$1/$3;}
|'('exp')' {$$=$2;}
```

```
|NUM  {$$=$1;}
;
%%
int main()
{
        printf("Enter the expression\n");
        yyparse();
}
int yyerror()
{
        printf("\nInvalid Expression");
        exit(0);
}
```

Execution Steps:

```
yacc –d 1b.y
lex 1b.l
cc lex.yy.c y.tab.c –ll
. /a.out
```