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

**LEX PART**

%{

#include "y.tab.h"

#include <stdlib.h>

extern int yylval;

%}

%%

[0-9]+ {yylval=atoi(yytext);

return NUM;}

. {return yytext[0];}

**YACC PART**

%{

#include<stdio.h>

int valid=1;

%}

%token NUM

%left '+' '-'

%left '\*' '/'

%%

Stmt: expr{if(valid)

{ printf("Result=%d\n",$$);}

}

expr: expr '+' expr { $$=$1+$3;}

| expr '- 'expr { $$=$1-$3;}

| expr '\*' expr { $$=$1\*$3;}

| expr '/' expr { if($3==0)

{ valid=0;printf("Divide by zero error\n"); }

else

$$=$1/$3;}

| '(' expr')' { $$=$2;}

| NUM {$$=$1;}

;

%%

2+3 \*5

|  |  |  |  |
| --- | --- | --- | --- |
| S🡪E   * E + E * E+E \*E * E +E \*num * E+ num\*num * num+num\*num | RSF | Handle | Action |
| num +num \*num  ( 2+ 3\*5) | Num | E🡪 num  $$ = $1  E = 2 |
| E +num \*num  (2 + num \*num) | num | E🡪 num  $$ = $1  E = 3 |
| E + E \*num  (2 + 3 \*num) | num | E🡪 num  $$ = $1  E = 5 |
| E + E \* E  (2 + 3 \*5) | E \*E | E = E \*E  $$ = $1\* $3  =3\*5=15 |
|  | E+E  (2+15) | E+E | E = E + E  $$ = $1+ $3  = 2+15=17 |
|  | E  (17) | E | S=E  $$ = $1  S =17 |
|  | S |  | S= 17 |

void main()

{

printf("Enter an Arithmetic Expression:\n");

yyparse();

if(valid==1)

printf("Expression is valid\n");

}

int yyerror()

{

printf("Invalid Expression\n");

exit(0);

}

**OUTPUT**

[root@localhost ss]# lex lab1b.l

[root@localhost ss]# yacc -d lab1b.y

[root@localhost ss]# cc lex.yy.c y.tab.c -ll

[root@localhost ss]# ./a.out

Enter an Arithmetic Expression:

2+3\*6

Result=20

Expression is valid

[root@localhost ss]# ./a.out

Enter an Arithmetic Expression:

2+3/0

Divide by zero error

[root@localhost ss]# ./a.out

Enter an Arithmetic Expression:

(2\*8)/2

Result=8

Expression is valid

2. Develop, Implement and Execute a program using YACC tool to recognize all strings ending with b preceded by n a’s using the grammar anb (note: input n value).

LEX PART

%{

#include "y.tab.h"

%}

%%

a {return A;}

b {return B;}

\n {return 0;}

. {return yytext[0];}

YACC PART

%{

#include <stdio.h>

int aCount=0,n;

%}

%token A

%token B

%%

S : X B { if (aCount<n || aCount>n)

{

YYFAIL();

}

}

X : X T | T

T : A { aCount++;}

;

%%

int main()

{ printf("Enter the value of n \n");

scanf("%d",&n);

printf("Enter the string\n");

yyparse();

printf("Valid string\n");

}

int YYFAIL()

{

printf("Invalid count of 'a'\n");

exit(0);

}

int yyerror()

{

printf("Invalid string\n");

exit(0);

}

|  |  |  |  |
| --- | --- | --- | --- |
| S🡪X B | RSF | Handle | Action |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

OUTPUT

[root@localhost ss]# lex lab2.l

[root@localhost ss]# yacc -d lab2.y

[root@localhost ss]# cc lex.yy.c y.tab.c -ll

[root@localhost ss]# ./a.out

Enter the value of n

2Enter the string

aab

Valid string

[root@localhost ss]# ./a.out

Enter the value of n

2Enter the string

aaab

Invalid Count of 'a'

[root@localhost ss]# ./a.out

Enter the value of n

2Enter the string

aabb

Invalid string

6(b) Write YACC program to recognize valid identifier, operators and keywords in the given text (C program) file.

LEX PART

%{

#include <stdio.h>

#include "y.tab.h"

%}

%%

int|char|bool|float|void|for|do|while|if|else|return|void|main {printf("keyword is %s\n",yytext);return KEY;}

[+|-|\*|/|=|<|>] {printf("operator is %s\n",yytext);return OP;}

[a-zA-Z][\_a-zA-Z0-9]\* {printf("identifier is %s\n",yytext);return ID;}

. ;

YACC PART

%{

#include <stdio.h>

#include <stdlib.h>

int id=0, key=0, op=0;

%}

%token ID KEY OP

%%

S: ID S { id++; }

| KEY S { key++; }

| OP S {op++;}

| ID { id++; }

| KEY { key++; }

| OP { op++;}

;

%%

extern FILE \*yyin;

void main(int argc ,char\*\* argv)

{

yyin = fopen(argv[1],"r");

yyparse();

printf("Keywords = %d\nIdentifiers = %d\nOperators = %d\n", key,id, op);

}

void yyerror()

{

printf("Not valid");

}

[root@localhost ~]# gedit input6b.c

void main()

{

int amount, \_sum = 0;

float roi=12.3;

int total12 = amount \* roi;

}

OUTPUT

[root@localhost ss]# lex lab6b.l

[root@localhost ss]# yacc -d lab6b.y

[root@localhost ss]# cc lex.yy.c y.tab.c -ll

[root@localhost ~]# ./a.out input6b.c

keyword is void

keyword is main

keyword is int

identifier is amount

identifier is \_sum

operator is =

keyword is float

identifier is roi

operator is =

keyword is int

identifier is total12

operator is =

identifier is amount

operator is \*

identifier is roi

Keywords = 5

Identifiers = 6

operators = 4