# LPCC LAB

**ASSIGNMENT NO: 4**

**Name : Pranav Sarda**

**Roll : 321047**

**GR : 21810773**

**Batch : TY Comp A2**

**Aim: Write a program to**

## Evaluate an arithmetic expression

1. **Built-in Functions**
2. **Variables using YACC specification.**
3. **Evaluate an arithmetic expression:**

**Source code:**

Lex File:

%{

#include<stdio.h> #include "y.tab.h" extern intyylval;

%}

%%

[0-9]+ {

yylval=atoi(yytext);

return NUMBER;

}

[\t] ;

[\n] return 0;

. return yytext[0];

%%

intyywrap()

{

return 1;

}

YACC FILE:

%{

#include<stdio.h> int flag=0;

%}

%token NUMBER

%left '+' '-'

%left '\*' '/' '%'

%left '(' ')'

%%

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;}

;

%%

void main()

{

printf("\nEnter Arithmetic Expression:\n"); yyparse();

if(flag==0)

printf("\nEntered arithmetic expression is Valid\n\n");

}

void yyerror()

{

printf("\nEntered arithmetic expression is Invalid\n\n"); flag=1;}

## Output:

Enter Any Arithmetic Expression : 5+7-3\*3/9\*100

Result = -88

Entered arithmetic expression is Valid

Enter Any Arithmetic Expression : 5+7-

Entered arithmetic expression is Invalid

1. **Built in Functions:**

## Source code:

Lex File:

%{

#include<stdio.h>

#include"y.tab.h"

#include<string.h>

int l;

%}

%%

[0-9]+ { yylval.dval = atoi(yytext); return NUMBER; }

[\t] ;

pow return POW;

sqrt return SQRT;

log return LOG;

strlen return STR;

sin return SIN;

cos return COS;

tan return TAN;

[\n] return 0;

. return yytext[0];

%%

int yywrap()

{

return 1;

}

YACC File:

%{

#include<math.h> //Run gcc with -lm flag

#include<stdio.h>

#include<string.h>

%}

%union

{

double dval;

}

%token <dval> NUMBER

%token SQRT

%token STR

%token LOG

%token POW

%token SIN

%token COS

%token TAN

%type <dval> E

%%

exp: E{

printf("\nResult=%lf\n",$1);

return 0;

}

E:SQRT'('NUMBER')' {$$ = sqrt($3);}

|STR NUMBER {$$= $2;}

|LOG'('NUMBER')' {$$ = log($3);}

|POW'('NUMBER','NUMBER')' {$$ = pow($3,$5);}

|SIN'('NUMBER')' {$$ = sin($3\*(3.14159265/180));}

|COS'('NUMBER')' {$$ = cos($3\*(3.14159265/180));}

|TAN'('NUMBER')' {$$ = tan($3\*(3.14159265/180));}

;

%%

void main()

{

printf("\nEnter A Valid Function : ");

yyparse();

}

int yyerror(char \*errormsg)

{

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

exit(1);

}

OUTPUT:

Enter A Valid Function : sqrt(7)

Result=2.645751

Enter A Valid Function : pow(5,3)

Result=125.000000

c) Variables using YACC specification:

## Source code:

Lex File:

%{

#include "y.tab.h"

%}

DIGIT [0-9]

TEXT [A-Za-z]

NUMBER [0-9]+

REAL [0-9]\*[.][0-9]+

CHARACTER \'.\'

IDENTIFIER [A-Za-z$\_]({DIGIT}|{TEXT}|\_|$)\*

ASSN =

SC ;

COMMA ,

%%

boolean { printf("%s\t==> BOOLEAN\n",yytext); return(BOOLEAN); }

int { printf("%s\t==> INT\n",yytext); return(INT); }

char { printf("%s\t==> CHAR\n",yytext); return(CHAR); }

float { printf("%s\t==> FLOAT\n",yytext); return(FLOAT); }

true|false { printf("%s\t==> BIT\n",yytext); return(BIT); }

{IDENTIFIER} { printf("%s\t==> IDENTIFIER\n",yytext); return(IDENTIFIER); }

{NUMBER} { printf("%s\t==> NUMBER\n",yytext); return(NUMBER); }

{REAL} { printf("%s\t==> REAL\n",yytext); return(REAL); }

{CHARACTER} { printf("%s\t==> CHARACTER\n",yytext); return(CHARACTER); }

{ASSN} { printf("%s\t==> ASSIGNMENT\n",yytext); return(ASSN); }

{SC} { printf("%s\t==> SEMICOLON\n",yytext); return(SC); }

{COMMA} { printf("%s\t==> COMMA\n",yytext); return(COMMA); }

. ;

%%

int yywrap()

{

return 1;

}

YACC File:

%{

FILE \*yyin;

int yylex();

void yyerror(const char \*s);

%}

%name parse

%token BOOLEAN INT CHAR FLOAT BIT NUMBER CHARACTER REAL IDENTIFIER ASSN SC COMMA

%%

s : sboolean | sint | schar | sfloat

sboolean : BOOLEAN blist SC { printf("Boolean Variable Declared\n"); } | BOOLEAN blist SC s { printf("Boolean Variable Declared\n"); } ;

blist : bvar | bvar COMMA blist ;

bvar : IDENTIFIER | IDENTIFIER ASSN BIT

sint : INT ilist SC { printf("Integer Variable Declared\n"); } | INT ilist SC s { printf("Integer Variable Declared\n"); } ;

ilist : ivar | ivar COMMA ilist ;

ivar : IDENTIFIER | IDENTIFIER ASSN NUMBER

schar : CHAR clist SC { printf("Character Variable Declared\n"); } | CHAR clist SC s { printf("Character Variable Declared\n"); } ;

clist : cvar | cvar COMMA clist ;

cvar : IDENTIFIER | IDENTIFIER ASSN CHARACTER

sfloat : FLOAT flist SC { printf("Float Variable Declared\n"); } | FLOAT flist SC s { printf("Float Variable Declared\n"); } ;

flist : fvar | fvar COMMA flist ;

fvar : IDENTIFIER | IDENTIFIER ASSN REAL

%%

void yyerror(const char \*s )

{

fprintf(stderr, "ERROR: %s\n",s);

}

int main()

{

yyin = fopen("input.java","r");

yyparse();

fclose(yyin);

return 0;

}

## Output:

**int ==> INT**

**a ==> IDENTIFIER**

**= ==> ASSIGNMENT**

**10 ==> NUMBER**

**, ==> COMMA**

**b ==> IDENTIFIER**

**, ==> COMMA**

**\_c\_ ==> IDENTIFIER**

**= ==> ASSIGNMENT**

**1 ==> NUMBER**

**; ==> SEMICOLON**

**float ==> FLOAT**

**$b ==> IDENTIFIER**

**= ==> ASSIGNMENT**

**100.9 ==> REAL**

**; ==> SEMICOLON**

**boolean ==> BOOLEAN**

**flag ==> IDENTIFIER**

**= ==> ASSIGNMENT**

**true ==> BIT**

**; ==> SEMICOLON**

**char ==> CHAR**

**\_ch ==> IDENTIFIER**

**= ==> ASSIGNMENT**

**'A' ==> CHARACTER**

**; ==> SEMICOLON**

**Character Variable Declared**

**Boolean Variable Declared**

**Float Variable Declared**

**Integer Variable Declared**