Name: Mihir Thakkar

Class: TY A

Roll No: 59

SRN:201901267

CD ASSIGNMENT-4

Design a YACC and corresponding LEX specification to compute the value of an expression. Consider arithmetic, trigonometric, 1/x, sqrt(x), x ^y etc. operators.

CODE:

(LEX CODE)

```
%option noyywrap
%{
#include"cal.tab.h"
#include<math.h>
#include<stdlib.h>
%}
%%
([0-9]+|([0-9]*\.[0-9]+)([eE][-+]?[0-9]+)?) {yylval.dval=atof(yytext);return
NUMBER;}
dec |
CONVERT {return CONVERT;}
pt |
POWTEN {return POWTEN;}
pi |
PI {return PI;}
exp |
EXP {return EXP;}
log |
LOG {return LOG;}
ln  |
LN  { return LN;}
sin |
```

```
SIN {return SIN;}
cos |
COS {return COS;}
tan |
TAN {return TAN;}
asin |
ASIN {return ASIN;}
acos |
ACOS {return ACOS;}
atan |
ATAN {return ATAN;}
sq |
SQUARE {return SQUARE;}
cube |
CUBE {return CUBE;}
sqrt |
SQRT {return SQRT;}
cbrt |
CBRT {return CBRT;}
fact |
FACTORIAL {return FACTORIAL;}
ceil |
CEILING {return CEILING;}
floor |
FLOOR {return FLOOR;}
cel |
CELSIUS {return CELSIUS;}
fah |
FAHRENHEIT {return FAHRENHEIT;}
kel |
KELVIN {return KELVIN;}
```

```
mem {return MEM;}
[\t];
\$ return 0;
\n|. return yytext[0];
%%
```

(YACC CODE)

```
%{
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
double memvar;

float pi(int x)
{
        float ans;
        ans = pow(3.14159,x);
        return ans;
}

float kel(float k)
{
        float celsius,fahrenheit;
        celsius=k-273.15;
        fahrenheit=celsius*9/5+32;
        printf("\nFahrenheit = %.2f  and Celsius = %.2f\n", fahrenheit, celsius);
    exit(0);
}

float fah(float f)
{
```

```
    float celsius,kelvin;
    celsius=(f-32)*5/9;
    kelvin=celsius-273.15;
    printf("\nCelsius = %.2f  and Kelvin = %.2f\n", celsius, kelvin);
    exit(0);
}

float cel(float c)
{
    float kelvin,fahrenheit;
    kelvin=c+273.15;
    fahrenheit=c*9/5+32;
    printf("\nFahrenheit = %.2f and Kelvin = %.2f\n",fahrenheit, kelvin);
    exit(0);
}

int call(int num)
{

   printf("\nBinary number = ");
   convert(num, 2);
   printf("\n");
   printf("\nOctal number = ");
   convert(num, 8);
   printf("\n");
   printf("\nHexadecimal number = ");
   convert(num, 16);
   printf("\n\n");
   exit(0);
}
```

```
void convert(int num, int base)
{
	int rem = num%base;

	if(num==0)
		return;
	convert(num/base, base);

	if(rem < 10)
		printf("%d", rem);
	else
		printf("%c", rem-10+'A' );
}

float factorial(int n)
{
  int c;
  float result = 1;

  for (c = 1; c <= n; c++)
    result = result * c;

  return result;
}

%}
%union
{
double dval;
}
%token<dval>NUMBER
```

```
%token<dval>MEM

%token POWTEN PI EXP LOG LN SIN COS TAN ASIN ACOS ATAN SQUARE CUBE SQRT CBRT FACTORIAL
CEILING FLOOR CONVERT CELSIUS FAHRENHEIT KELVIN

%left '-' '+'

%left '*' '/' '%'

%right '^'

%left POWTEN PI EXP LOG LN SIN COS TAN ASIN ACOS ATAN SQUARE CUBE SQRT CBRT FACTORIAL
CEILING FLOOR CONVERT CELSIUS FAHRENHEIT KELVIN

%nonassoc UMINUS

%type<dval>expression

%%

start:statement'\n'

|start statement'\n'

;


statement:MEM'='expression {memvar=$3;}

| expression {printf("\nThe result is = %g\n",$1);

main();

}

;

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

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

| expression '*' expression {$$=$1*$3;}

| expression '/' expression

{ if($3==0)

yyerror("Cannot divide by zero!");

else

$$=$1/$3;

}

|expression '%' expression {$$=fmod($1,$3);}

|expression '^' expression {$$=pow($1,$3);}

;
```

```
expression:'-'expression %prec UMINUS{$$=-$2;}
|'('expression')'{$$=$2;}
|'['expression']'{$$=$2;}
|POWTEN expression {$$=pow(10,$2);}
|PI expression {$$=pi($2);}
|EXP expression {$$=exp($2);}
|LOG expression
{if ($2<0)
{printf("Enter a positve integer!\n");
exit(0);}
else
$$=log($2)/log(10);}
|LN expression
{if ($2<0)
{printf("Enter a positve integer!\n");
exit(0);}
else
$$=log($2);}
|SIN expression {$$=sin($2*3.14159/180);}
|COS expression {$$=cos($2*3.14159/180);}
|TAN expression
{ if ($2==90 || $2==270)
{printf("Result = Not defined\n");
exit(0);}
else
$$=tan($2*3.14159/180);}
|ASIN expression {$$=round(asin($2)*180/3.14159);}
|ACOS expression {$$=round(acos($2)*180/3.14159);}
|ATAN expression {$$=round(atan($2)*180/3.14159);}
|SQUARE expression {$$=$2*$2;}
|CUBE expression {$$=$2*$2*$2;}
```

```
|SQRT expression {$$=sqrt($2);}
|CBRT expression {$$=cbrt($2);}
|FACTORIAL expression
{ if ($2<0)
{printf("Enter a positve integer!\n");
exit(0);}
else
$$=factorial((int)$2);}
|CEILING expression {$$=ceil($2);}
|FLOOR expression {$$=floor($2);}
|CONVERT expression
{if ($2<0)
{printf("Enter a positve integer!\n");
exit(0);}
else
$$=call($2);}
|CELSIUS expression {$$=cel($2);}
|FAHRENHEIT expression {$$=fah($2);}
|KELVIN expression {$$=kel($2);}
|NUMBER {$$=$1;}
|MEM {$$=memvar;}
;
%%
int main()
{
printf("\nEnter your mathematical expression: ");
yyparse();
}
int yyerror(char *error)
{
printf("%s\n",error);}
```

OUTPUT:

```
D:\SEM VI\Assignments\CD\Ass4>flex cal.l

D:\SEM VI\Assignments\CD\Ass4>bison -d cal.y

D:\SEM VI\Assignments\CD\Ass4>gcc lex.yy.c cal.tab.c
cal.y: In function 'call':
cal.y:46:5: warning: implicit declaration of function 'convert' [-Wimplicit-function-declaration]
     convert(num, 2);
     ^~~~~~~
cal.y: At top level:
cal.y:57:6: warning: conflicting types for 'convert'
 void convert(int num, int base)
      ^~~~~~~
cal.y:46:5: note: previous implicit declaration of 'convert' was here
     convert(num, 2);
     ^~~~~~~
cal.tab.c: In function 'yyparse':
cal.tab.c:775:16: warning: implicit declaration of function 'yylex' [-Wimplicit-function-declaration]
 # define YYLEX yylex ()
                ^
cal.tab.c:1420:16: note: in expansion of macro 'YYLEX'
       yychar = YYLEX;
                ^~~~~
cal.y:103:1: warning: implicit declaration of function 'main' [-Wimplicit-function-declaration]
 main();
 ^~~~
cal.y:111:1: warning: implicit declaration of function 'yyerror' [-Wimplicit-function-declaration]
 yyerror("Cannot divide by zero!");
 ^~~~~~~

D:\SEM VI\Assignments\CD\Ass4>a.exe
```

```
D:\SEM VI\Assignments\CD\Ass4>a.exe

Enter your mathematical expression: 2+sin(90)

The result is = 3

Enter your mathematical expression: tan(90)
Result = Not defined

D:\SEM VI\Assignments\CD\Ass4>a.exe

Enter your mathematical expression: dec(11)

Binary number = 1011

Octal number = 13

Hexadecimal number = B


D:\SEM VI\Assignments\CD\Ass4>a.exe

Enter your mathematical expression: 1.5*2.5/3

The result is = 1.25

Enter your mathematical expression: 100/0
Cannot divide by zero!

The result is = 100
```