Name: Dhroov Makwana
Gr no.: 22010538
Roll no.: 321042
Batch: A2

Assignment 4

Aim: Write a program to evaluate an arithmetic expression, built-in functions and variables using YACC specifications Write a program

4 A) To evaluate an arithmetic expression using YACC tool

4 B) To evaluate or check built-in functions using YACC tool

4 C) To recognize valid variable name using YACC tool.

Theory:

YACC stands for Yet Another Compiler Compiler. YACC provides a tool to produce a parser for a given grammar. YACC is a program designed to compile a LALR (1) grammar.

It is used to produce the source code of the syntactic analyzer of the language produced by LALR (1) grammar. The input of YACC is the rule or grammar and the output is a C program.

A YACC program consists of three sections: Declarations, Rules and Auxiliary functions.

DECLARATIONS

%%

RULES

%%

AUXILIARY FUNCTIONS

i) **Declarations:** The declarations section consists of two parts: (i) C declarations and (ii) YACC declarations . The C Declarations are delimited by %{ and %}. This part consists of all the declarations required for the C code written in the Actions section and the Auxiliary functions section. YACC copies the contents of this section into the generated y.tab.c file without any modification.

ii) **Rules:** A rule in a YACC program comprises of two parts (i) the production part and (ii) the action part. A rule in YACC is of the form:

production_head : production_body {action in C } ;

iii) **Auxiliary Functions**: The Auxiliary functions section contains the definitions of three mandatory functions main(), yylex() and yyerror(). You may wish to add your own functions (depending on the the requirement for the application) in the y.tab.c file. Such functions are written in the auxiliary functions section. The main() function must invoke yyparse() to parse the input.

For Output Files:

• The output of YACC is a file named y.tab.c

• If it contains the main() definition, it must be compiled to be executable.

• Otherwise, the code can be an external function definition for the function int yyparse()

• If called with the –d option in the command line, Yacc produces as output a header file y.tab.h with all its specific definition (particularly important are token definitions to be included, for example, in a Lex input file).

• If called with the –v option, Yacc produces as output a file y.output containing a textual description of the LALR(1) parsing table used by the parser. This is useful for tracking down how the parser solves conflicts.

Compilation Steps of Program

- yacc –d filename.y
- lex filename.l
- gcc lex.yy.c y.tab.c
- ./a.out

Code 4 A):

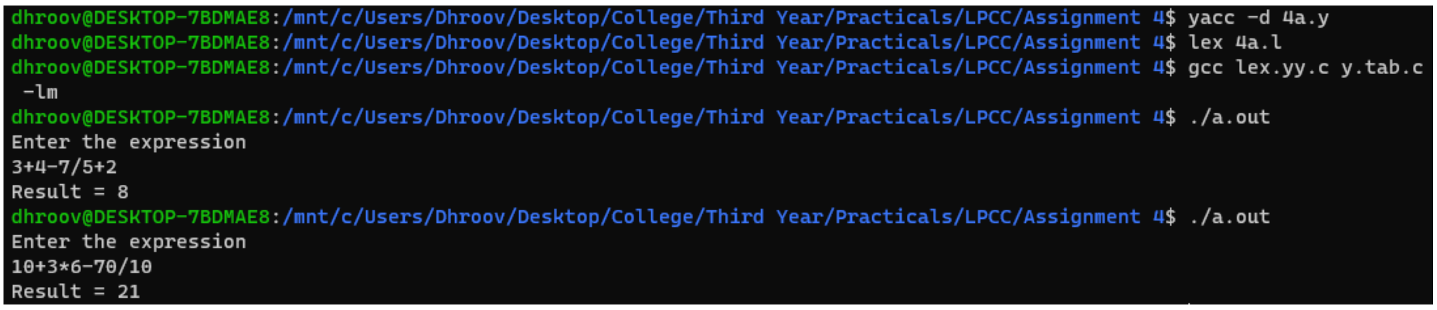
LEX FILE

```
%{
    #include "y.tab.h"
    int yylval;
%}

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

[a-zA-Z]+  { return ID; }
[ \t]+ ;

\n { return 0; }
.  { return yytext[0]; }

%%
```

YACC FILE

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

%token num ID

%left '+' '-'
%left '*' '/'

%%

E : T  {
    printf("Result = %d\n", $$);
    return 0;
}

T :
    T '+' T { $$ = $1 + $3; }
    | T '-' T { $$ = $1 - $3; }
    | T '*' T { $$ = $1 * $3; }
    | T '/' T
    {
    if($3==0)
        {
            printf("Cannot divide by 0");
            exit(0);
        }
    else
    {
        $$ = $1 / $3;
    }
    }
| '-' num { $$ = -$2; }
| '-' ID { $$ = -$2; }
| '(' T ')' { $$ = $2; }
| num { $$ = $1; }
| ID { $$ = $1; };
%%

int yywrap()
{
    return 1;
}

int main() {
    printf("Enter the expression\n");
    yyparse();
}


void yyerror() {
    printf("\nExpression is invalid\n");
}
```

Output 4 A)



```
dhroov@DESKTOP-7BDMAE8:/mnt/c/Users/Dhroov/Desktop/College/Third Year/Practicals/LPCC/Assignment 4$ yacc -d 4a.y
dhroov@DESKTOP-7BDMAE8:/mnt/c/Users/Dhroov/Desktop/College/Third Year/Practicals/LPCC/Assignment 4$ lex 4a.l
dhroov@DESKTOP-7BDMAE8:/mnt/c/Users/Dhroov/Desktop/College/Third Year/Practicals/LPCC/Assignment 4$ gcc lex.yy.c y.tab.c
 -lm
dhroov@DESKTOP-7BDMAE8:/mnt/c/Users/Dhroov/Desktop/College/Third Year/Practicals/LPCC/Assignment 4$ ./a.out
Enter the expression
3+4-7/5+2
Result = 8
dhroov@DESKTOP-7BDMAE8:/mnt/c/Users/Dhroov/Desktop/College/Third Year/Practicals/LPCC/Assignment 4$ ./a.out
Enter the expression
10+3*6-70/10
Result = 21
```

Code 4 B):

LEX FILE

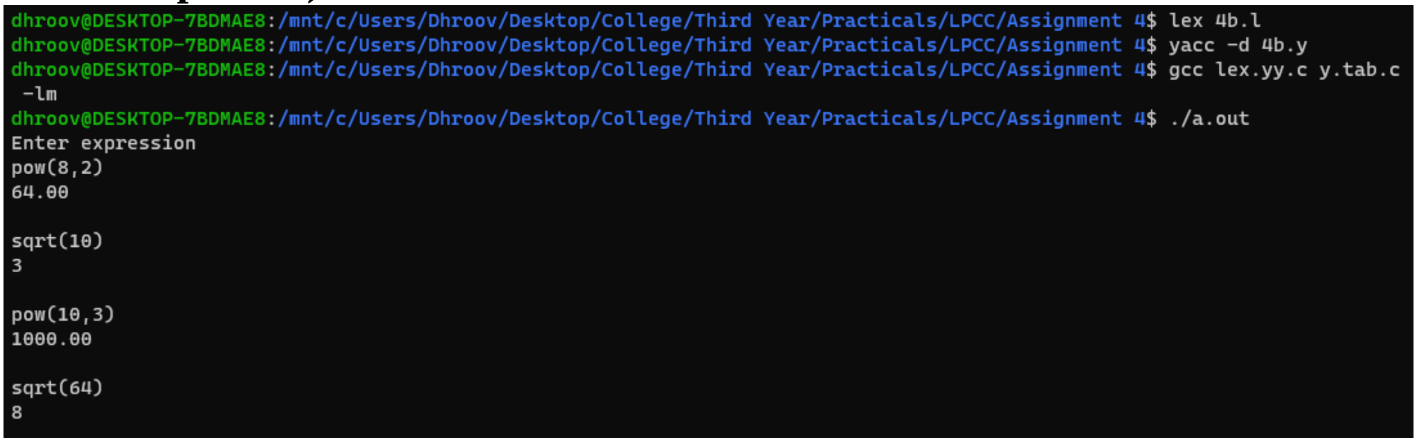
```
%{
    #include"y.tab.h"
%}

%%
```

```
sqrt return sqrt_fun;
[0-9]+ {yylval = atoi(yytext); return num;}
[(] return OB;
[)] return CB;
pow return POW;
, return comma;
[\n] return NL;

%%
```

YACC FILE

```
%{
    #include<math.h>
    #include<stdio.h>
    #include<string.h>
    int yylval;
    void yyerror();
    int yylex();
%}

%token OB CB num sqrt_fun alpha POW comma NL;

%%
START: s NL START | s NL;
s: sqrt_fun OB num CB { $$ = sqrt($3);
printf("%d\n\n",$$);} |
POW OB num comma num CB {printf("%.2f\n\n",pow($3,$5));}

;


%%

int yywrap()
{
    return 1;
}

int main()
{
    printf("Enter expression\n");
    yyparse();
}

void yyerror()
{
    printf("Invalid\n");
}
```

Output 4 B)



```
dhroov@DESKTOP-7BDMAE8:/mnt/c/Users/Dhroov/Desktop/College/Third Year/Practicals/LPCC/Assignment 4$ lex 4b.l
dhroov@DESKTOP-7BDMAE8:/mnt/c/Users/Dhroov/Desktop/College/Third Year/Practicals/LPCC/Assignment 4$ yacc -d 4b.y
dhroov@DESKTOP-7BDMAE8:/mnt/c/Users/Dhroov/Desktop/College/Third Year/Practicals/LPCC/Assignment 4$ gcc lex.yy.c y.tab.c
 -lm
dhroov@DESKTOP-7BDMAE8:/mnt/c/Users/Dhroov/Desktop/College/Third Year/Practicals/LPCC/Assignment 4$ ./a.out
Enter expression
pow(8,2)
64.00

sqrt(10)
3

pow(10,3)
1000.00

sqrt(64)
8
```

Code 4 C)

LEX FILE

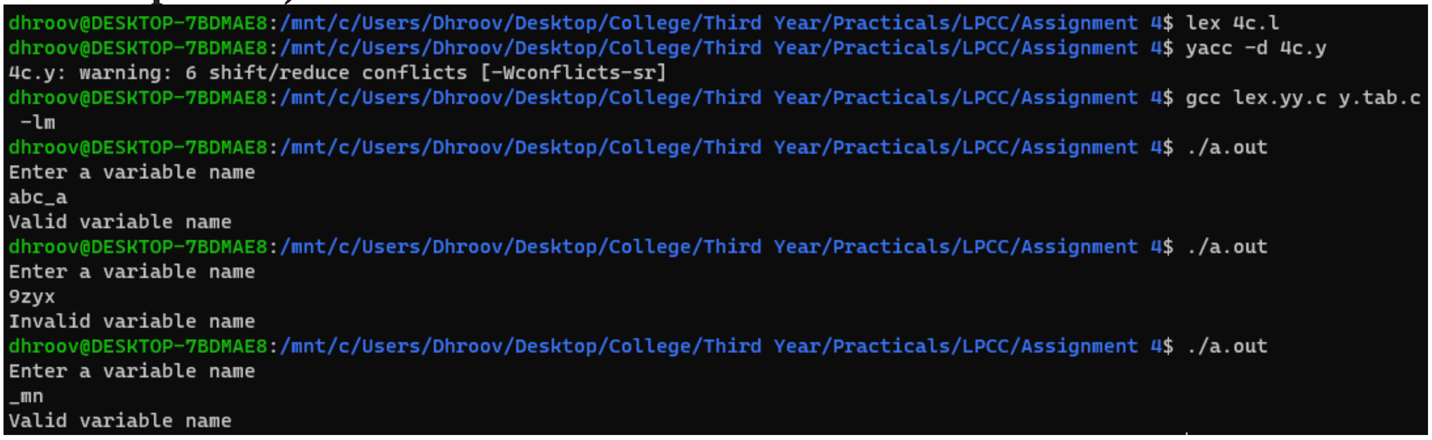
```
%{
    #include "y.tab.h"
%}

%%
[a-zA-Z] return alpha;

[0-9] return num;
"_" return underscore;
\n return 0;

%%
```

YACC FILE

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

%token alpha num underscore;

%%

a: a num
   | a alpha
   | alpha
   | a underscore a
   | underscore a
    ;
```

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

int main()
{
    printf("Enter a variable name\n");
    yyparse();
    printf("Valid variable name\n");
}

void yyerror()
{
    printf("Invalid variable name\n");
    exit(0);
}
```

Output 4 C)



```
dhroov@DESKTOP-7BDMAE8:/mnt/c/Users/Dhroov/Desktop/College/Third Year/Practicals/LPCC/Assignment 4$ lex 4c.l
dhroov@DESKTOP-7BDMAE8:/mnt/c/Users/Dhroov/Desktop/College/Third Year/Practicals/LPCC/Assignment 4$ yacc -d 4c.y
4c.y: warning: 6 shift/reduce conflicts [-Wconflicts-sr]
dhroov@DESKTOP-7BDMAE8:/mnt/c/Users/Dhroov/Desktop/College/Third Year/Practicals/LPCC/Assignment 4$ gcc lex.yy.c y.tab.c
 -lm
dhroov@DESKTOP-7BDMAE8:/mnt/c/Users/Dhroov/Desktop/College/Third Year/Practicals/LPCC/Assignment 4$ ./a.out
Enter a variable name
abc_a
Valid variable name
dhroov@DESKTOP-7BDMAE8:/mnt/c/Users/Dhroov/Desktop/College/Third Year/Practicals/LPCC/Assignment 4$ ./a.out
Enter a variable name
9zyx
Invalid variable name
dhroov@DESKTOP-7BDMAE8:/mnt/c/Users/Dhroov/Desktop/College/Third Year/Practicals/LPCC/Assignment 4$ ./a.out
Enter a variable name
_mn
Valid variable name
```

Conclusion: I have successfully implemented yacc programs to evaluate a arithmetic expression, evaluate in-built functions and to recognize a valid variable name.