CS6612 – Compiler Lab

Ex no : 5 **Name : Sreedhar V**

Date : 10.03.2021 **Reg no: 185001161**

- Implementation of Desk Calculator using Yacc Tool

Specification

Write Lex program to recognize relevant tokens required for the Yacc parser to implement desk calculator. Write the Grammar for the expression involving the operators namely, + , - ,* , / , ^ , (,).Precedence and associativity has to be preserved. Yacc is available as a command in linux. Thegrammar should have non terminals E, Op and a terminal id.

Code

(lex file)

```
%{
#include "calc.tab.h"
#include <stdio.h>
#include<string.h>
void yyerror(char *);
int yylval;
%}
num [0-9]+
tab "\t"|"\n"
GT ">="
LT "<="
LS "<<"
RS ">>"
LAND "&&"
LOR "||"
EQ "=="
NQ "!="
%%
{num} {yylval=atoi(yytext);return NUM;}
{tab} {return 0;}
. return yytext[0];
{GT} {return GT;}
{LT} {return LT;}
{LS} {return LS;}
{RS} {return RS;}
```

```
{LAND} {return LAND;}
{LOR} {return LOR;}
{NQ} {return NQ;}
{EQ} {return EQ;}
%%
int yywrap(void){return 1;}
```

(yacc file)

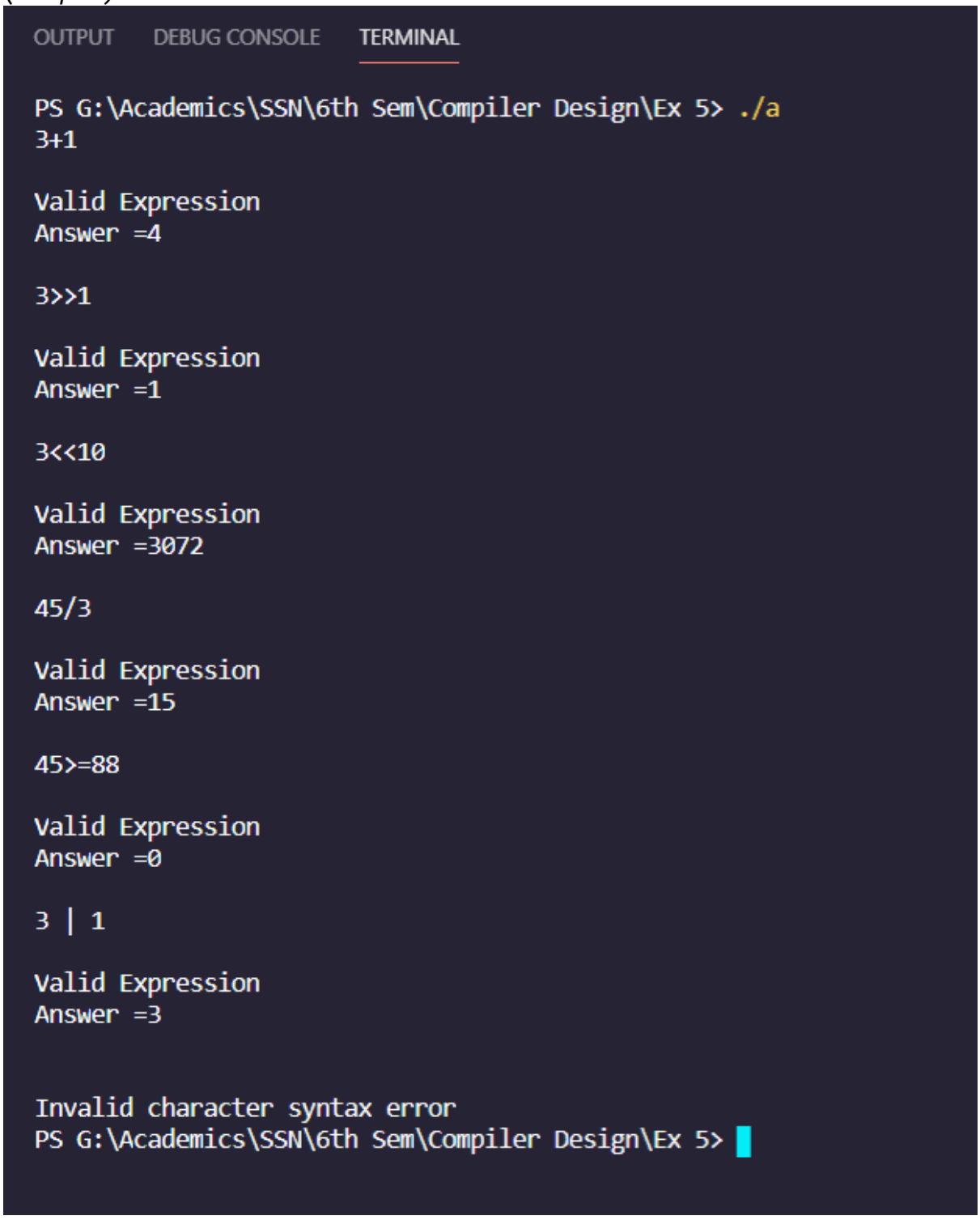
```
%{
#include <stdio.h>
int flag=0;
int yyerror(char *er);
int yylex(void);
#include <math.h>
#include<stdlib.h>
%}
%token NUM
%token LS
%token RS
%token GT
%token LT
%token LAND
%token EQ
%token NQ
%token LOR
%left LOR
%left LAND
%left '|'
%left '&'
%left EQ NQ
%left '>' GT
%left '<' LT
%left LS RS
%left '+' '-'
%left '*' '^' '/' '%'
%left '(' ')'

%%
P : E {if(!flag){printf("\nValid Expression\n");}printf("Answer =%d\n\n
",$$);return 0;};
E : NUM { $$ = $1;}
  | E '+' E {$$ = $1 + $3; }
  | E '-' E {$$ = $1 - $3; }
  | E '*' E {$$ = $1 * $3; }
  | E '/' E {$$ = $1 / $3; }
```

```
  | E '%' E {$$ = $1 % $3; }
  | E '^' E {$$ = pow($1,$3); }
  | '(' E ')' {$$ = $2 ; }
  | E GT E {$$ = $1>= $3; }
  | E '>' E {$$ = $1> $3; }
  | E '<' E {$$ = $1< $3; }
  | E LT E {$$ = $1<= $3; }
  | E RS E {$$ = $1>>$3; }
  | E LS E {$$ = $1<< $3; }
  | E '&' E {$$ = $1 & $3; }
  | E '|' E {$$ = $1 | $3; }
  | E LAND E {$$ = $1 && $3; }
  | E LOR E {$$ = $1 || $3; }
%%
int main()
{
    while(1)
    {
        yyparse();
    }


    return 0;
}
int yyerror(char *er)
{
    flag=1;
    printf("\nInvalid character %s\n",er);
    exit(0);
}
```

(Output)



Learning Outcome:

- I've learnt how to implement the calculator considering all its precedence and associativity while execution.

- I’ve learnt the basic syntax of the yacc program and how to implement the grammar rules in c code.
- I’ve learnt the associativity and precedence of the operators and how to evaluate whether the given expression is a valid one or not.
- I’ve leant how the lex program sends the token based on its syntax and yacc program evaluates the stream of tokens based on the given grammar rules and produces the result.