**Exercise8**

Design a LALR Bottom-up parser for the given language

**Save this color part as parser.l**

%{

#include<stdio.h>

#include "y.tab.h"

%}

%%

[0-9]+ {yylval.dval=atof(yytext);

return DIGIT;

}

\n|. return yytext[0];

%%

Save this part as parser.y

%{

#include<stdio.h>

%}

%union

{

double dval;

}

%token <dval> DIGIT

%type <dval> expr

%type <dval> term

%type <dval> factor

%%

line: expr '\n' {

printf("%g\n",$1);

}

;

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

| term

;

term: term '\*' factor {$$=$1 \* $3 ;}

| factor

;

factor: '(' expr ')' {$$=$2 ;}

| DIGIT

;

%%

int main()

{

yyparse();

}

yyerror(char \*s)

{

printf("%s",s);

}

**Output:**

$lex parser.l

$yacc –d parser.y

$cc lex.yy.c y.tab.c –ll –lm

$./a.out

2+3

5.0000