LP LAB ASSIGNMENT (Remove Redundant Parentheses)

Ravi Jain (411764)

Lex.h

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;
#define A 65
#define B 66
#define C 67
#define D 68
#define E 69
#define F 70
#define G 71
#define H 72
#define I 73
#define OP1 10 //plus
#define OP2 11 //minus
#define OP3 21 //multipication
#define OP4 22 //division
#define OP5 31 //'('
#define OP6 32 //')'
char symbol_table[15][2];
char input_buffer[15];
int no_tokens;
char buffer;
bool flag=true;
int sym_ptr=-1;

void install_token(char token) {
		bool check=true;
		if(sym_ptr!=-1){
				for(int i=0; i<sym_ptr; i++) {
						if(symbol_table[i][0]==token) {
								check=false;
						}
				}
		}
		if(check){
				symbol_table[++sym_ptr][0]=token;
				switch(token) {
						case 'A': symbol_table[sym_ptr][1]='1';
									break;
						case 'B': symbol_table[sym_ptr][1]='1';
									break;
						case 'C': symbol_table[sym_ptr][1]='1';
									break;
						case 'D': symbol_table[sym_ptr][1]='1';
									break;
						case 'E': symbol_table[sym_ptr][1]='1';
									break;
						case 'F': symbol_table[sym_ptr][1]='1';
									break;
						case 'G':symbol_table[sym_ptr][1]='1';
									break;
						case 'H':symbol_table[sym_ptr][1]='1';
									break;
						case 'I': symbol_table[sym_ptr][1]='1';
									break;
						case '+': symbol_table[sym_ptr][1]='2';
									break;
						case '-':symbol_table[sym_ptr][1]='2';
									break;
						case '*': symbol_table[sym_ptr][1]='2';
									break;
						case '/': symbol_table[sym_ptr][1]='2';
									break;
						case '(':symbol_table[sym_ptr][1]='3';
									break;
						case ')':symbol_table[sym_ptr][1]='3';
									break;
				}
		}
```

```
}

void print_symbol_table() {
        cout << "\n\n=================Symbol Table=================\n\n";
        cout << "SYMBOL | TYPE\n";
        for(int i=0; i<=sym_ptr; i++){
                cout << symbol_table[i][0] << "     |     " << symbol_table[i][1] << "\n";
        }
        cout << "\n\n==========================================\n\n";
}

void input() {
        cin >> input_buffer;
        int cpos=0, pos=0;
        while(input_buffer[pos]!='\0') {
                if(input_buffer[pos]!=' ') {
                        char temp=input_buffer[pos];
                        if(int(temp)>=97 && int(temp)<=122) {
                                temp=char(int(temp)-32);
                        }
                        if(temp=='A' || temp=='B' || temp=='C' || temp=='D' || temp=='E' || temp=='F' || temp=='G' || temp=='H' ||
temp=='I' || temp=='+' || temp=='-' || temp=='/' || temp=='*' || temp=='(' || temp==')') {
                                input_buffer[cpos]=temp;
                                cpos++;
                        }
                }
                pos++;
        }
        if(pos!=cpos){
                input_buffer[cpos]='\0';
        }
}

int tokens_check(int *to_return){
        no_tokens=-1;
        int input_buffer_length=strlen(input_buffer);
        while(input_buffer_length) {
                buffer=input_buffer[++no_tokens];
                install_token(buffer);
                switch(buffer) {
                        case 'A': to_return[no_tokens]=A;
                                  break;
                        case 'B': to_return[no_tokens]=B;
                                  break;
                        case 'C': to_return[no_tokens]=C;
                                  break;
                        case 'D': to_return[no_tokens]=D;
                                  break;
                        case 'E': to_return[no_tokens]=E;
                                  break;
                        case 'F': to_return[no_tokens]=F;
                                  break;
                        case 'G': to_return[no_tokens]=G;
                                  break;
                        case 'H': to_return[no_tokens]=H;
                                  break;
                        case 'I': to_return[no_tokens]=I;
                                  break;
                        case '+': to_return[no_tokens]=OP1;
                                  break;
                        case '-': to_return[no_tokens]=OP2;
                                  break;
                        case '*': to_return[no_tokens]=OP3;
                                  break;
                        case '/': to_return[no_tokens]=OP4;
                                  break;
                        case '(': to_return[no_tokens]=OP5;
                                  break;
                        case ')': to_return[no_tokens]=OP6;
                                  break;
                        default: flag=false;
                }
                if(!flag) {
                        cout << "\n\nLex Error\n\n";
                        return 0;
                }
                input_buffer_length--;
        }
        return 1;
```

```
}

int return_no_token() {
        return no_tokens;
}
```

parser.h

```
#include "lex.h"
#include <stdexcept>
#define LENGTH 100

struct Node {
        string data;
        char op;
};

int tree_lvl=0;

void tree_lvl_print() {
        for(int i=0; i<tree_lvl; i++){
                cout<< "---";
        }
}

void LookAhead();
char look_up(int);
void ID(string*, char*);
void RetainParantheses(string*, char*, string*, char*);
void paren(string*, char*);
void factor(string*, char*);
void term(string*, char*);
void express(string*, char*);
string start_line();

int tokens[LENGTH];
int no_token;
int lookahead;
int curr_position=-1;

void init_tokens() {
        cout << "\ninit_token\n";
        for(int i=0; i<LENGTH; i++){
                tokens[i]=-1;
        }
}

void LookAhead() {
        cout << "LookAhead\n";
        ++curr_position;
        cout << "curr_position: " <<curr_position <<"\n";
}

char look_up(int x) {
        cout << "look_up\n";
        char temp;
        switch(x) {
                case 65 :temp='A';
                                break;
                case 66 :temp='B';
                                break;
                case 67 :temp='C';
                                break;
                case 68 :temp='D';
                                break;
                case 69 :temp='E';
                                break;
                case 70 :temp='F';
                                break;
                case 71 :temp='G';
                                break;
                case 72 :temp='H';
                                break;
                case 73 :temp='I';
                                break;
                case 10 :temp='+';
                                break;
```

```
                        case 11:temp='-';
                                        break;
                        case 21:temp='*';
                                        break;
                        case 22:temp='/';
                                        break;
                        case 31:temp='(';
                                        break;
                        case 32:temp=')';
                                        break;
                        default: temp=' ';
                }
                return temp;
}

string start_line() {
                cout << "start_line\n";
                LookAhead();
                string var;
                char var_op;
                express(&var, &var_op);
                cout  <<"return start_line:";
                if(curr_position>no_token) {
                        cout << "RETURN\n\n\n";
                        tree_lvl_print();
                        tree_lvl--;
                        return var;
                }
                cout << "\n\n\n";
                tree_lvl_print();
                tree_lvl--;
                return var;
}

void express(string *to_return, char *to_return_op){
                tree_lvl++;
                tree_lvl_print();
                cout << "express\n";
                string Lside_var;
                char Lside_var_op;
                term(&Lside_var, &Lside_var_op);
                int op=tokens[curr_position];
                if (op != 10 && op != 11) {
                        *to_return=Lside_var;
                        *to_return_op=Lside_var_op;
                        tree_lvl_print();
                        tree_lvl--;
                        cout << "express Lside Return\n";
                  return;
                }
                cout << "+|- Encountered\n";
    LookAhead();
                string Rside_var;
                char Rside_var_op;
                term(&Rside_var, &Rside_var_op);
    *to_return=Lside_var + look_up(op) + Rside_var;
    *to_return_op = look_up(op);
                tree_lvl_print();
                tree_lvl--;
    cout << "express Return\n";
}

void term(string *to_return, char *to_return_op) {
                tree_lvl++;
                tree_lvl_print();
                cout << "term\n";
                string Lside_var;
                char Lside_var_op;
                factor(&Lside_var, &Lside_var_op);
                *to_return=Lside_var;
                *to_return_op=Lside_var_op;
                if(curr_position>no_token) {
                        tree_lvl_print();
                        tree_lvl--;
                        cout << "\nRETURN\n";
                  return;
                }
                int op=tokens[curr_position];
                if(op!=21 && op!=22){
                        tree_lvl_print();
                        tree_lvl--;
                        cout<< "term Lside Return\n";
                  return;
                }
                cout << "*|/ Encountered\n";
                LookAhead();
                string Rside_var;
                char Rside_var_op;
                factor(&Rside_var, &Rside_var_op);
                bool a=false, b=false;
                string new_var_a;
                char new_var_a_op;
                string new_var_b;
                char new_var_b_op;
                if(Lside_var_op == '+' || Lside_var_op == '-') {
                        a=true;
                        RetainParantheses(&Lside_var, &Lside_var_op, &new_var_a, &new_var_a_op);
                }
                if(Rside_var_op == '+' || Rside_var_op == '-'){
                        b=true;
                        RetainParantheses(&Rside_var, &Rside_var_op, &new_var_b, &new_var_b_op);
                }
    *to_return_op = look_up(op);
    if(a && b) {
                *to_return = new_var_a + look_up(op) + new_var_b;
    }
    else if(a && !b) {
                *to_return = new_var_a + look_up(op) + Rside_var;
    }
    else {
                *to_return = Lside_var + look_up(op) + new_var_b;
    }
                tree_lvl_print();
                tree_lvl--;
    cout << "term Return\n";
}

void factor(string *to_return, char *to_return_op){
                tree_lvl++;
                tree_lvl_print();
                cout << "factor\n";
                int curr_token=tokens[curr_position];
                if(curr_token==31){
                        paren(to_return, to_return_op);
                }
                else if(curr_token!=31 && curr_token!=32 && curr_token!=10 && curr_token!=11 && curr_token!=21 && curr_token!=22){
                        ID(to_return, to_return_op);
                }
                tree_lvl_print();
                tree_lvl--;
                cout << "factor Return\n";
}

void ID(string *to_return, char *to_return_op) {
                tree_lvl++;
                tree_lvl_print();
                cout << "ID\n";
                char temp=look_up(tokens[curr_position]);
                *to_return=temp;
                *to_return_op=' ';
                LookAhead();
                tree_lvl_print();
                tree_lvl--;
                cout << "ID Return\n";
}

void paren(string *to_return, char *to_return_op) {
                tree_lvl++;
                tree_lvl_print();
                cout << "paren\n";
                LookAhead();
                express(to_return, to_return_op);
                if(tokens[curr_position] != 32) {
                        throw std::invalid_argument( "Expected ')'\n" );
                }
                tree_lvl_print();
                tree_lvl--;
                cout << "paren Return\n";
```

```
		LookAhead();
}

void RetainParantheses(string *var, char *op, string *to_return, char *to_return_op) {
		tree_lvl++;
		tree_lvl_print();
		cout << "RetainParantheses\n";
		*to_return="("+*var+")";
		*to_return_op=*op;
		tree_lvl_print();
		tree_lvl--;
		cout << "RetainParantheses Return\n";
}


void print_tokens() {
		cout << "\nNumber of tokens are: " << no_tokens << "\n";
		cout << "\n================= Tokens =================\n\n";
		for(int i=0; i<=no_token; i++) {
				cout<< tokens[i] << "  " << "\n";
		}
		cout << "\n========================================\n";
}
```

main.cpp

```
#include "parser.h"

int main(){
		input();
		init_tokens();
		if(tokens_check(tokens)){
				no_token=return_no_token();
				print_tokens();
				print_symbol_table();
				cout<< "\nLex work is done!\n";
		}
		else {
				cout<< "\nInvalid input string\n";
		}
		cout << "\nRESULT:\n\n" << start_line() << "\n\n";
		return 0;
}
```

SDT

```
=========SDT===============

  E -> E+T | E-T | T

E()			-->			T()T1()
T1()		-->			+T()T1()
T1()		-->			-T()T1()
T1()		-->			e


  T -> F | T*F | T/F

T()			-->			F()T2()
T2()		-->			*F()T2()
T2()		-->			/F()T2()
T2()		-->			e


  F ->		ID

F()		-->			ID()
F()			-->		'(' E() ')'


  ID -> A | B |.....|I

ID()		-->		A
ID()		-->			B
```

.
.
.
ID()   -->   I

LEX STD:

lex.l

```
%{
#include"y.tab.h"
void yyerror(char *);
%}

%%
[ \t\n]          {return 0;}
[A-Za-z]         {yylval = *yytext; return VAR;}
[-]              {yylval= *yytext; return OP1;}
[+]              {yylval= *yytext; return OP2;}
[*]              {yylval= *yytext; return OP3;}
[/]              {yylval= *yytext; return OP4;}
[(]              {yylval= *yytext; return OP5;}
[)]              {yylval= *yytext; return OP6;}
.                {ECHO;   yyerror ("unexpected character");
                                          }
%%

int yywrap (void){
        return 1;
}

void yyerror(char *s) {
   fprintf(stderr, "line %d: %s\n", yylineno, s);
}
```

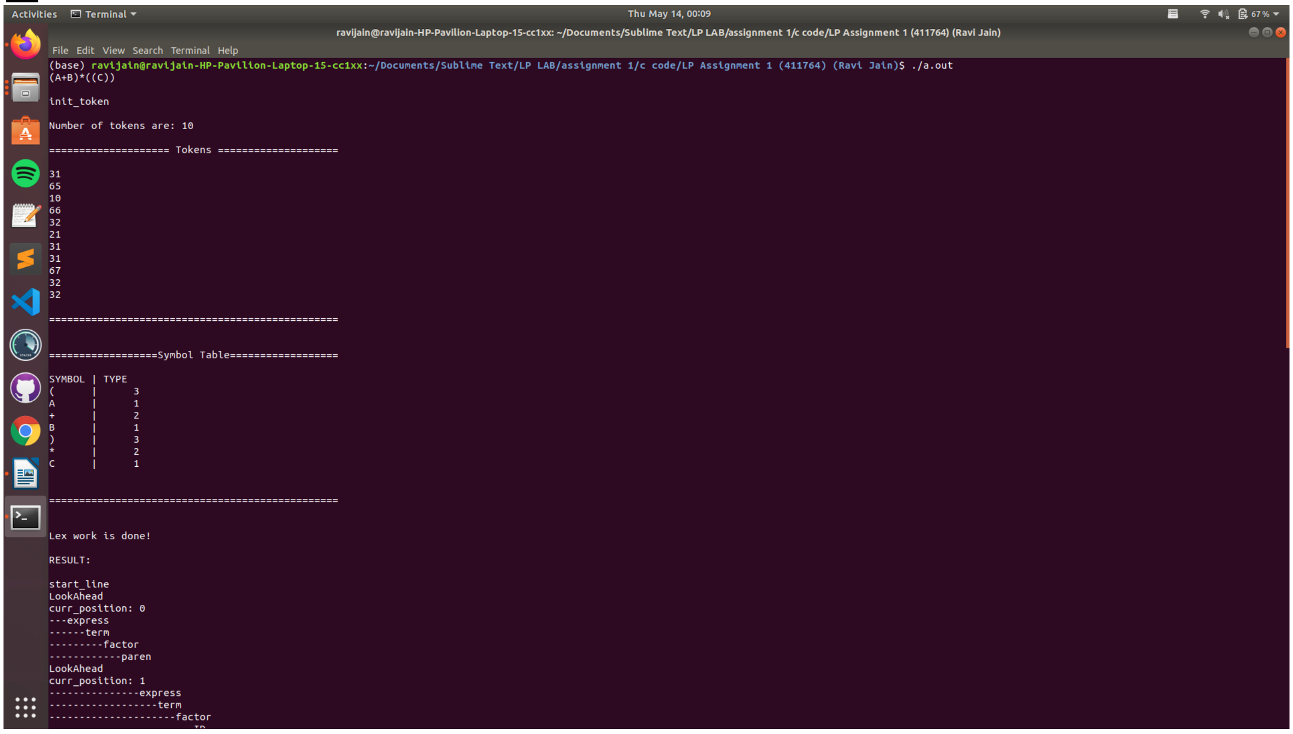
parser.l

```
%{
#include <ctype.h>
#include <stdio.h>
#include <string.h>
int yylex(void);
void yyerror(char *);
%}

 // %union {char *str;}
%start line
 // %type <str> exp term factor id
%token VAR OP1 OP2 OP3 OP4 OP5 OP6


%%

line     :         exp
         {printf ("1a %c\n", $1);}
           ;

exp      :  OP5    OP5 exp OP6 OP6                       {printf("removed '((exp))'\n");}
                   |        OP5 exp OP6                  {printf("removed '(exp)'\n");}
                   |  exp OP2  OP5 exp OP2 exp OP6       {printf("removed 'a+(c+b)\n");}
                   |  exp OP2  OP5 exp OP1 exp OP6       {printf("removed 'a+(c-b)'\n");}
                   |        exp OP1 OP5 exp OP2 exp OP6  {printf("removed 'a-(c+b)'\n");}
                   |        exp OP1 OP5 exp OP1 exp OP6  {printf("removed 'a-(c- b)'\n");}
                   |        exp OP2 OP5 exp OP3 exp OP6  {printf("removed 'a+(c*b)'\n");}
                   |        exp OP1 OP5 exp OP3 exp OP6  {printf("removed 'a-(c*b)\n");}
                   |        exp OP2 OP5 exp OP4 exp OP6  {printf("removed 'a+(c/b)'\n");}
                   |        exp OP1 OP5 exp OP4 exp OP6  {printf("removed 'a-(c/b)'\n");}
                   |        exp OP3 OP5 exp OP3 exp OP6  {printf("removed 'a*(c*b)'\n");}
                   |        exp OP4 OP5 exp OP4 exp OP6  {printf("removed 'a/(c/b)'\n");}
                   |        exp OP3 OP5 exp OP4 exp OP6  {printf("removed 'a*(c/b)'\n");}
                   |        exp OP4 OP5 exp OP3 exp OP6  {printf("removed 'a/(c*b)'\n");}
                   |        term                         {printf("\n");}
                   |        exp OP2 term                 {printf(" +\n");}
```

```
        |       exp OP1 term                    {printf(" -\n");}

term    :       factor                          {printf("%c\n",$1);}
        |       term OP3 factor                 {printf("*\n");}
        |       term OP4 factor                 {printf(" /\n");}
        ;

factor  :       id                               {printf("%c\n",$1);}
        |       OP5 exp OP6                      {printf("4b ( %c) \n",$2);}
                ;
id      :       VAR                             {printf("%c\n",$1);}
        |       OP1                             {printf("%c\n",$1);}
        |       OP2                             {printf("%c\n",$1);}
        |       OP3                             {printf("%c\n",$1);}
        |       OP4                             {printf("%c\n",$1);}
        |       OP5                             {printf("%c\n",$1);}
        |       OP6                             {printf("%c\n",$1);}
                ;
%%

int main(void){return yyparse( );}
```

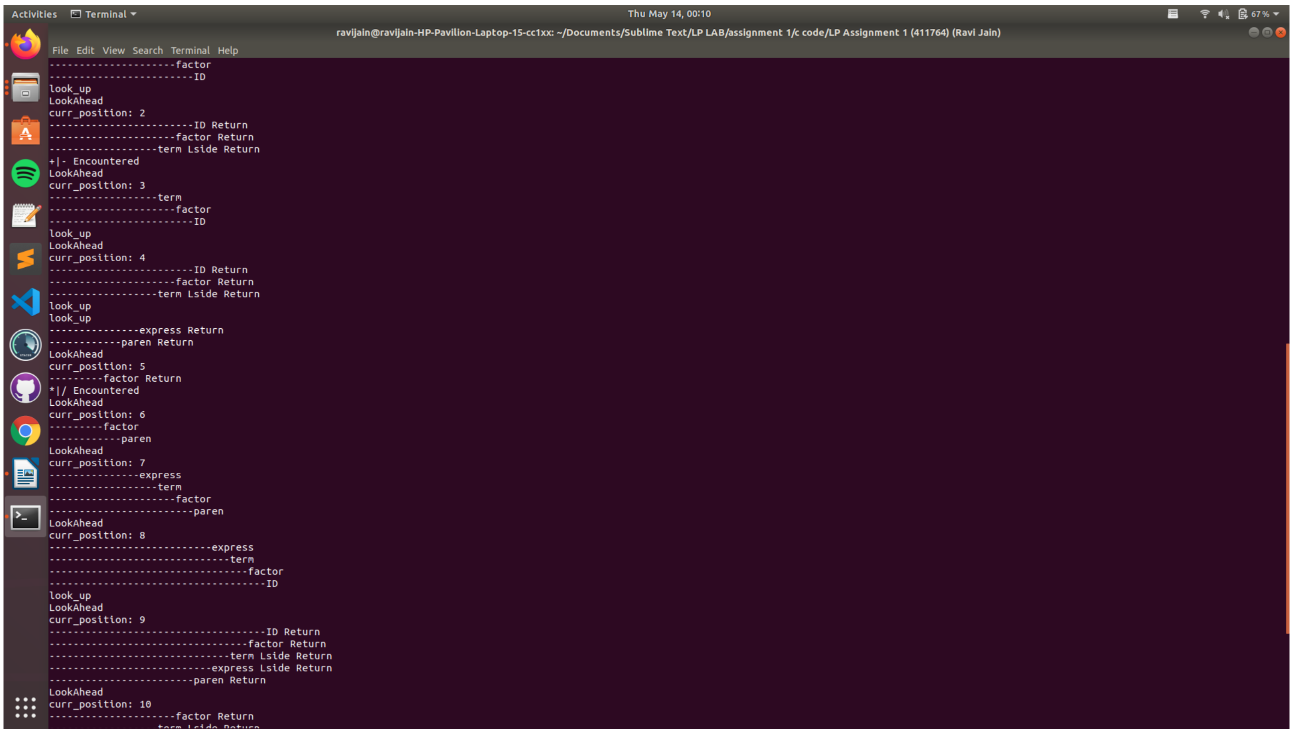
OUTPUT

1.

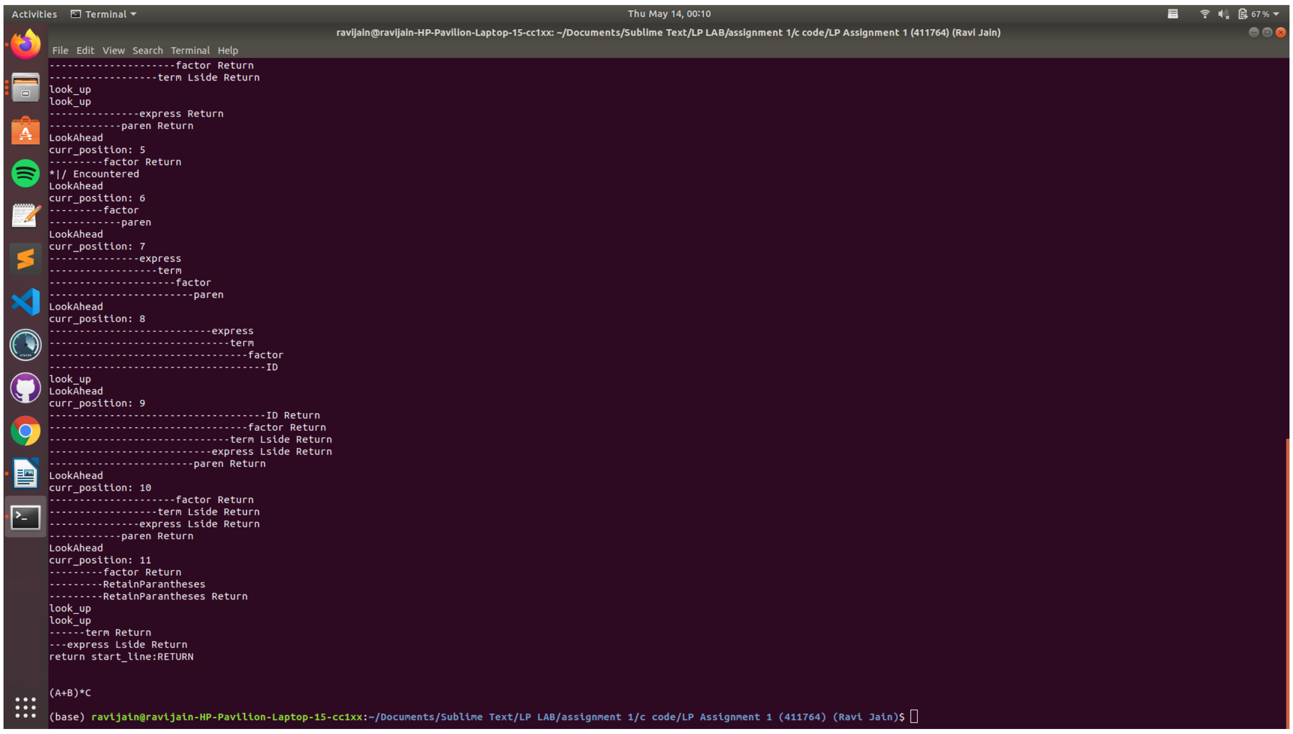


Activities
Terminal
Thu May 14, 00:09
67 %
ravijain@ravijain-HP-Pavilion-Laptop-15-cc1xx: ~/Documents/Sublime Text/LP LAB/assignment 1/c code/LP Assignment 1 (411764) (Ravi Jain)
File Edit View Search Terminal Help

```
(base) ravijain@ravijain-HP-Pavilion-Laptop-15-cc1xx:~/Documents/Sublime Text/LP LAB/assignment 1/c code/LP Assignment 1 (411764) (Ravi Jain)$ ./a.out
(A+B)*((C))

init_token

Number of tokens are: 10

=================== Tokens ===================

31
65
10
66
32
21
31
31
67
32
32

================================================


=================Symbol Table=================

SYMBOL | TYPE
(      |       3
A      |       1
+      |       2
B      |       1
)      |       3
*      |       2
C      |       1


================================================


Lex work is done!

RESULT:

start_line
LookAhead
curr_position: 0
---express
------term
---------factor
------------paren
LookAhead
curr_position: 1
---------------express
------------------term
---------------------factor
```

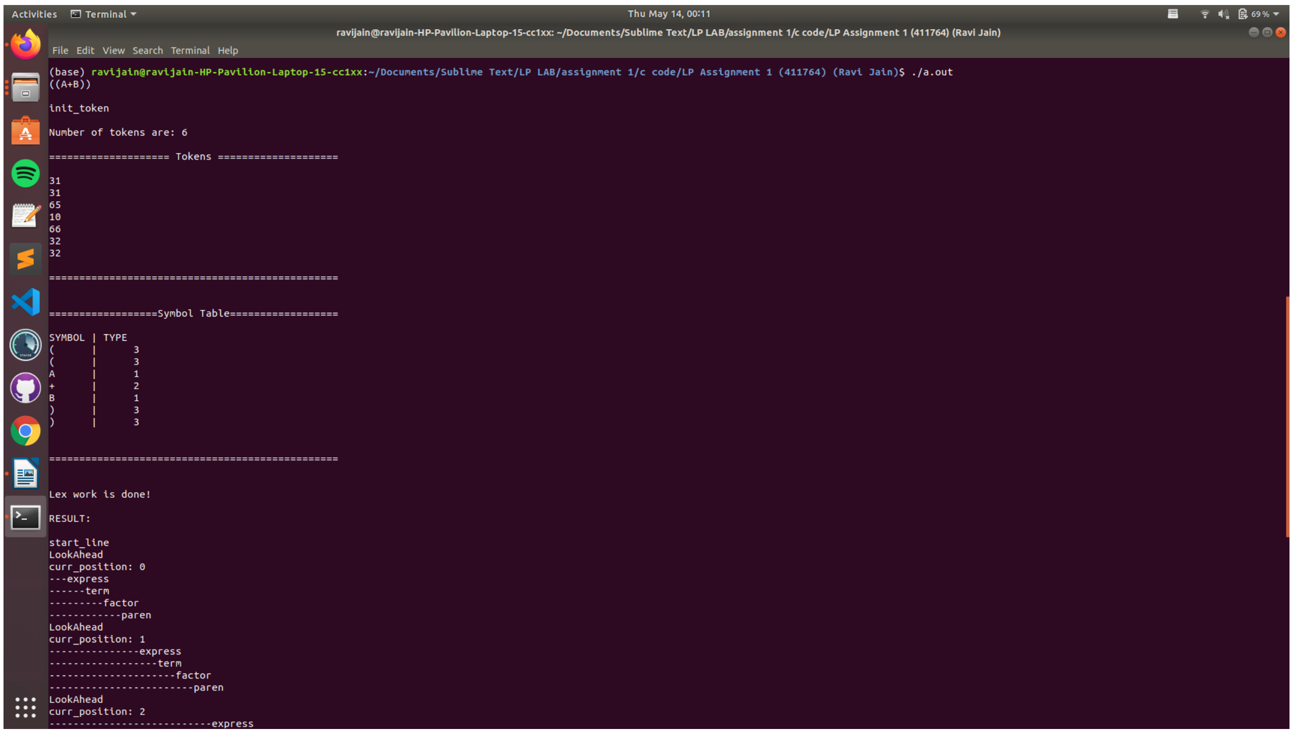

Activities
Terminal
Thu May 14, 00:10
67 %
ravijain@ravijain-HP-Pavilion-Laptop-15-cc1xx: ~/Documents/Sublime Text/LP LAB/assignment 1/c code/LP Assignment 1 (411764) (Ravi Jain)
File Edit View Search Terminal Help

```
---------------------factor
------------------------ID
look_up
LookAhead
curr_position: 2
------------------------ID Return
---------------------factor Return
------------------term Lside Return
+|- Encountered
LookAhead
curr_position: 3
------------------term
---------------------factor
------------------------ID
look_up
LookAhead
curr_position: 4
------------------------ID Return
---------------------factor Return
------------------term Lside Return
look_up
look_up
---------------express Return
------------paren Return
LookAhead
curr_position: 5
---------factor Return
*|/ Encountered
LookAhead
curr_position: 6
---------factor
------------paren
LookAhead
curr_position: 7
---------------express
------------------term
---------------------factor
------------------------paren
LookAhead
curr_position: 8
---------------------------express
------------------------------term
---------------------------------factor
------------------------------------ID
look_up
LookAhead
curr_position: 9
------------------------------------ID Return
---------------------------------factor Return
------------------------------term Lside Return
---------------------------express Lside Return
------------------------paren Return
LookAhead
curr_position: 10
---------------------factor Return
```

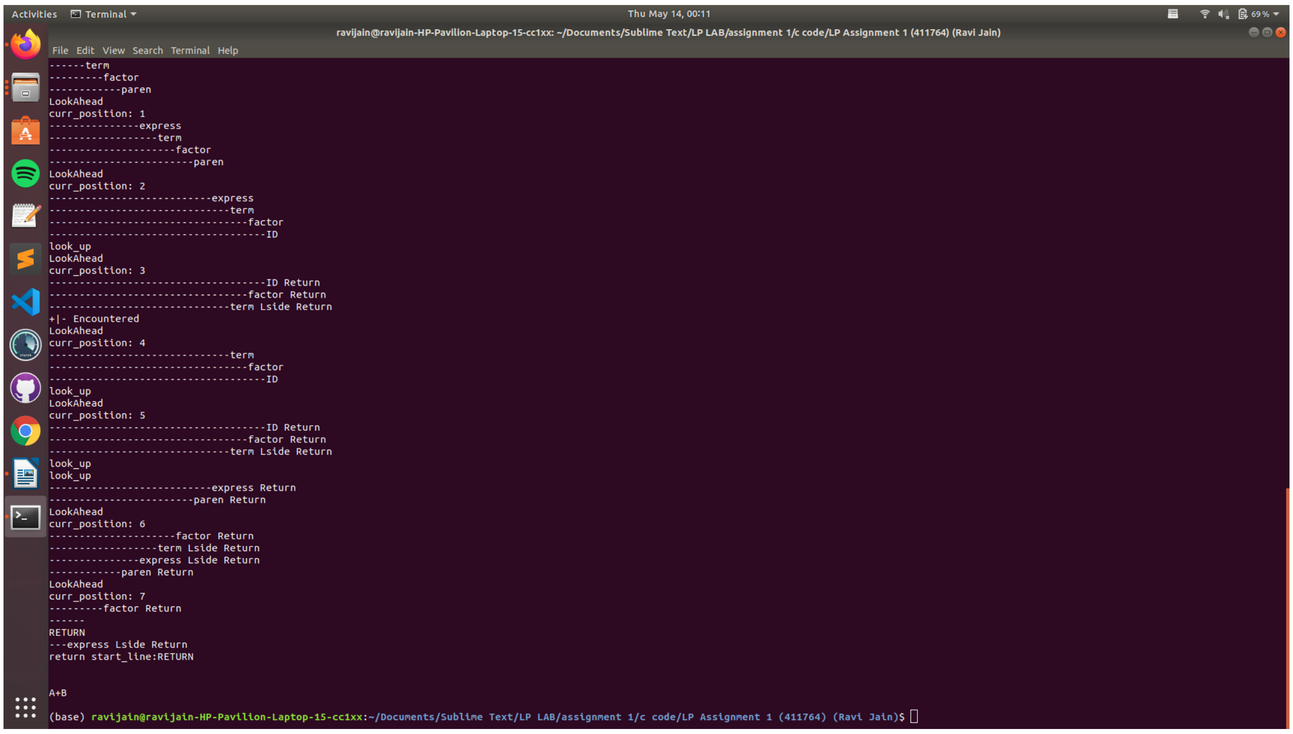

Activities
Terminal
Thu May 14, 00:10
67 %
ravijain@ravijain-HP-Pavilion-Laptop-15-cc1xx: ~/Documents/Sublime Text/LP LAB/assignment 1/c code/LP Assignment 1 (411764) (Ravi Jain)
File Edit View Search Terminal Help

```
---------------------factor Return
------------------term Lside Return
look_up
look_up
---------------express Return
------------paren Return
LookAhead
curr_position: 5
---------factor Return
*|/ Encountered
LookAhead
curr_position: 6
---------factor
------------paren
LookAhead
curr_position: 7
---------------express
------------------term
---------------------factor
------------------------paren
LookAhead
curr_position: 8
---------------------------express
------------------------------term
---------------------------------factor
------------------------------------ID
look_up
LookAhead
curr_position: 9
------------------------------------ID Return
---------------------------------factor Return
------------------------------term Lside Return
---------------------------express Lside Return
------------------------paren Return
LookAhead
curr_position: 10
---------------------factor Return
------------------term Lside Return
---------------express Lside Return
------------paren Return
LookAhead
curr_position: 11
---------factor Return
---------RetainParantheses
---------RetainParantheses Return
look_up
look_up
------term Return
---express Lside Return
return start_line:RETURN


(A+B)*C

(base) ravijain@ravijain-HP-Pavilion-Laptop-15-cc1xx:~/Documents/Sublime Text/LP LAB/assignment 1/c code/LP Assignment 1 (411764) (Ravi Jain)$ 
```

2.



```
(base) ravijain@ravijain-HP-Pavilion-Laptop-15-cc1xx:~/Documents/Sublime Text/LP LAB/assignment 1/c code/LP Assignment 1 (411764) (Ravi Jain)$ ./a.out
((A+B))

init_token

Number of tokens are: 6

==================== Tokens ====================

31
31
65
10
66
32
32

================================================


==================Symbol Table==================

SYMBOL | TYPE
(      |      3
(      |      3
A      |      1
+      |      2
B      |      1
)      |      3
)      |      3


================================================


Lex work is done!

RESULT:

start_line
LookAhead
curr_position: 0
---express
------term
---------factor
------------paren
LookAhead
curr_position: 1
---------------express
------------------term
---------------------factor
------------------------paren
LookAhead
curr_position: 2
---------------------------express
```



```
------term
---------factor
------------paren
LookAhead
curr_position: 1
---------------express
------------------term
---------------------factor
------------------------paren
LookAhead
curr_position: 2
---------------------------express
------------------------------term
---------------------------------factor
------------------------------------ID
look_up
LookAhead
curr_position: 3
------------------------------------ID Return
---------------------------------factor Return
------------------------------term Lside Return
+|- Encountered
LookAhead
curr_position: 4
------------------------------term
---------------------------------factor
------------------------------------ID
look_up
LookAhead
curr_position: 5
------------------------------------ID Return
---------------------------------factor Return
------------------------------term Lside Return
look_up
look_up
---------------------------express Return
------------------------paren Return
LookAhead
curr_position: 6
---------------------factor Return
------------------term Lside Return
---------------express Lside Return
------------paren Return
LookAhead
curr_position: 7
---------factor Return
------
RETURN
---express Lside Return
return start_line:RETURN


A+B

(base) ravijain@ravijain-HP-Pavilion-Laptop-15-cc1xx:~/Documents/Sublime Text/LP LAB/assignment 1/c code/LP Assignment 1 (411764) (Ravi Jain)$ 
```

Additional Info:

To complie the program: (Run the below commands with pwd as this folder) (In ubuntu)

1. g++ lex.h
2. g++ parser.h
3. g++ main.cpp
4. ./a.out

Rules for input:
1. Don't add any wide spaces.