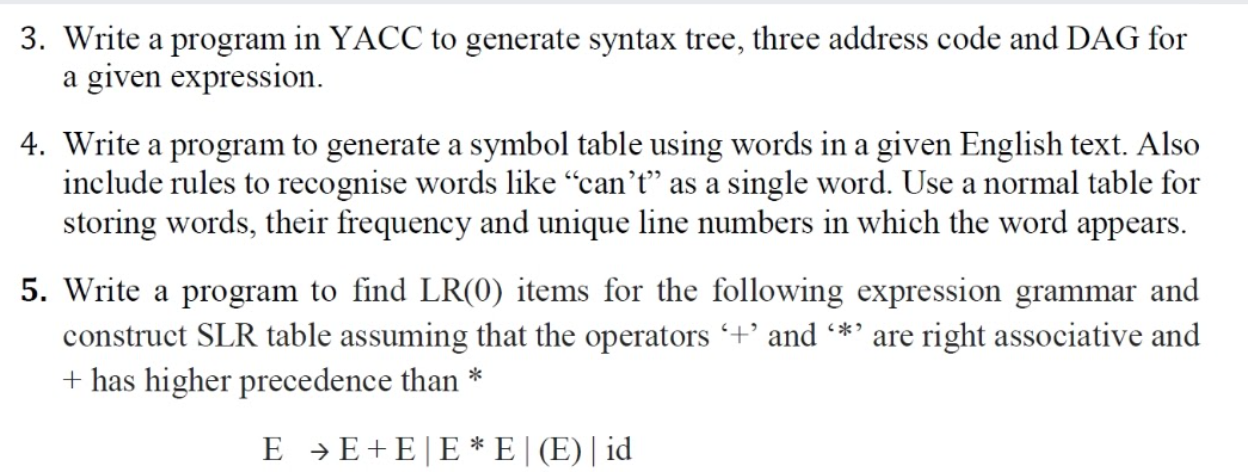
**LP ASSIGNMENT – 8**

**K.S.ABHIHKESHAV**

**21CSB0F03**

****

**3**) code:

Lex:

%{

# include<stdio.h>

# include "y.tab.h"

%}

%%

[a-zA-Z]+ {yylval.strval=strdup(yytext);return VAR;}

[0-9]+ {yylval.strval=strdup(yytext);return NUM;}

\t {}

\n {return NL;}

. {return yytext[0];}

%%

int yywrap(){

return 1;

}

Yacc:

%{

#include<stdio.h>

#include<stdlib.h>

%}

%union{

char\* strval;

}

%token NUM VAR NL

%left '+' '-'

%left '\*' '/'

%left ')' '('

%left '='

%%

exp: V NL{printf("%s",$<strval>1);exit(0);}

|E NL{printf("%s",$<strval>1);exit(0);}

;

V: VAR '=' E {char\* s=malloc(21+sizeof(char)\*(strlen($<strval>3)+strlen($<strval>1)));strcat(s,"( root: = lc: ");strcat(s,$<strval>1);strcat(s," rc: ");strcat(s,$<strval>3);strcat(s," )");$<strval>$=s;}

;

E:

|E '+' E {char\* s=malloc(21+sizeof(char)\*(strlen($<strval>3)+strlen($<strval>1)));strcat(s,"( root: + lc: ");strcat(s,$<strval>1);strcat(s," rc: ");strcat(s,$<strval>3);strcat(s," )");$<strval>$=s;}

|E '-' E {char\* s=malloc(21+sizeof(char)\*(strlen($<strval>3)+strlen($<strval>1)));strcat(s,"( root: - lc: ");strcat(s,$<strval>1);strcat(s," rc: ");strcat(s,$<strval>3);strcat(s," )");$<strval>$=s;}

|E '\*' E {char\* s=malloc(21+sizeof(char)\*(strlen($<strval>3)+strlen($<strval>1)));strcat(s,"( root: \* lc: ");strcat(s,$<strval>1);strcat(s," rc: ");strcat(s,$<strval>3);strcat(s," )");$<strval>$=s;}

|E '/' E {char\* s=malloc(21+sizeof(char)\*(strlen($<strval>3)+strlen($<strval>1)));strcat(s,"( root: / lc: ");strcat(s,$<strval>1);strcat(s," rc: ");strcat(s,$<strval>3);strcat(s," )");$<strval>$=s;}

|'(' E ')' {$<strval>$=$<strval>2;}

| NUM {$<strval>$=$<strval>1;}

| VAR {$<strval>$=$<strval>1;}

;

%%

int main(){

yyparse();

}

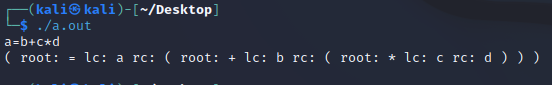
void yyerror(const\* msg){

printf("invalid string\n");

exit(0);

}

Output:



4)

Code:

#include <bits/stdc++.h>

using namespace std;

int main() {

unordered\_map<string,pair<int,string>> mp;

vector<string> str;

string s;

while(true)

{

getline(cin,s);

if(s=="$")

break;

else str.push\_back(s+" ");

}

string temp;

for(int i=0; i<str.size(); i++){

temp = str[i];

string word;

for(int j=0; j<temp.size(); j++){

if(temp[j]!= ' ' && temp[j]!='.'){

word += temp[j];

}

else if(temp[j]==' '){

//cout<<"word:"<<word<<" i:"<<i<<endl;

auto itr = mp.find(word);

if(itr ==mp.end()){

mp.insert({word,{1,to\_string(i+1)}});

}

else{

itr->second.first++;

itr->second.second = itr->second.second+" "+to\_string(i+1);

}

word ="";

}

}

}

for(auto i=mp.begin(); i!=mp.end(); i++){

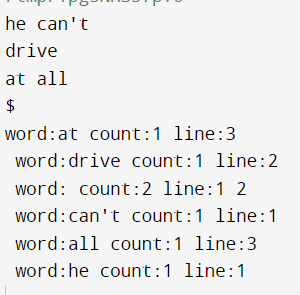
cout<<" word:"<<i->first<<" count:"<<i->second.first<<" line:"<<i->second.second<<endl;

}

return 0;

}

Output:



5)

Code:

#include <iostream>

#include<cmath>

#include<bits/stdc++.h>

#include<numeric>

#include<string>

using namespace std;

map<string,string> eqnum;

string table[20][7];

char symbol[]={'\*','+','(',')','i','$','E'};

struct node{

string arr[20];

int top=0;

};

node item[20];

int yy=0;

int createnode(node &x,node m,char a){

int flag=0;

for(int i=0;i<m.top;i++){

if(m.arr[i][m.arr[i][0]-'0']==a){

x.arr[x.top++]=m.arr[i];

x.arr[x.top-1][0]+=1;

if(x.arr[x.top-1][x.arr[x.top-1][0]-'0']=='E')

flag=1;

}

}

if(x.top==0)

return -2;

if(flag==1){

x.arr[x.top++]="1E+E#";

x.arr[x.top++]="1E\*E#";

x.arr[x.top++]="1(E)#";

x.arr[x.top++]="1i#";

}

for(int i=0;i<yy;i++){

if(item[i].top==x.top){

map<string,int> z;

for(int j=0;j<x.top;j++){

z[item[i].arr[j]]++;

}

int cc=0;

for(int j=0;j<x.top;j++){

if(z[x.arr[j]]==1)

cc++;

}

if(cc==x.top)

return i;

}

}

return -1;

}

void gen(node n,int num){

item[num]=n;

cout<<num<<endl;

for(int i=0;i<n.top;i++){

cout<<n.arr[i]<<endl;

}

cout<<endl;

for(int i=0;i<n.top;i++){

if(n.arr[i][n.arr[i][0]-'0']=='#'){

string x=eqnum[n.arr[i].substr(1,n.arr[i].length()-2)];

if(x=="acc")

table[num][5]+=x;

else{

table[num][0]+="r"+x;

table[num][1]+="r"+x;

table[num][3]+="r"+x;

table[num][4]+="r"+x;

table[num][5]+="r"+x;

}

}

}

for(int i=0;i<7;i++){

node temp1;

int x=createnode(temp1,n,symbol[i]);

if(x>=0){

if(i==6)

table[num][i]=char('0'+x);

else{

table[num][i]+="s";

table[num][i]+=char('0'+x);

}

}

else if(x==-1){

if(i==6)

table[num][i]=char('0'+yy);

else{

table[num][i]+="s";

table[num][i]+=char('0'+yy);

}

gen(temp1,yy++);

}

}

}

int main() {

node n;

eqnum["E"]="acc";

eqnum["E+E"]="1";

eqnum["E\*E"]="2";

eqnum["(E)"]="3";

eqnum["i"]="4";

for(int i=0;i<20;i++){

for(int j=0;j<7;j++){

table[i][j]="";

}

}

n.arr[n.top++]="1E#";

n.arr[n.top++]="1E+E#";

n.arr[n.top++]="1E\*E#";

n.arr[n.top++]="1(E)#";

n.arr[n.top++]="1i#";

cout<<"LR items"<<endl;

gen(n,yy++);

cout<<"\n\nstates\t\*\t+\t(\t)\ti\t$\tE"<<endl;

for(int i=0;i<yy;i++){

cout<<i<<"\t";

for(int j=0;j<7;j++){

if(table[i][j]=="")

cout<<" \t";

else

cout<<table[i][j]<<"\t";

}

cout<<"\n";

}

return 0;

}

**Output: rows numbers are jumbled but SLR TABLE IS CORRECT for the given numbering of LR(1) items numbers**

