**SAVEETHASCHOOLOF ENGINEERING**

**CSA1455**

**COMPILERDESIGN**

**LAB MANUAL**

**KISHORE G**

**192365051**

# Exp.No.1

**DevelopalexicalAnalyzertoidentifyidentifiers,constants,operatorsusingCprogram.**

**Program:** #include<stdio.h> #include<ctype.h> #include<string.h>inint main()

{

inti,ic=0,m,cc=0,oc=0,j;

charb[30],operators[30],identifiers[30],constants[30]; printf("enter the string : ");

scanf("%[^\n]s",&b); for(i=0;i<strlen(b);i++)

{

if(isspace(b[i]))

{

continue;

}

elseif(isalpha(b[i]))

{

identifiers[ic]=b[i]; ic++;

}

else if(isdigit(b[i]))

{ m=(b[i]-'0');

i=i+1; while(isdigit(b[i]))

{

m=m\*10+(b[i]-'0'); i++;

}

i=i-1; constants[cc]=m; cc++;

}

else

{

if(b[i]=='\*')

{

operators[oc]='\*'; oc++;

}

elseif(b[i]=='-')

{

operators[oc]='-'; oc++;

}

else if(b[i]=='+')

{

operators[oc]='+'; oc++;

}

else if(b[i]=='=')

{

operators[oc]='='; oc++;

}

}

}

printf("identifiers:"); for(j=0;j<ic;j++)

{

printf("%c",identifiers[j]);

}

printf("\nconstants:"); for(j=0;j<cc;j++)

{

printf("%d",constants[j]);

}

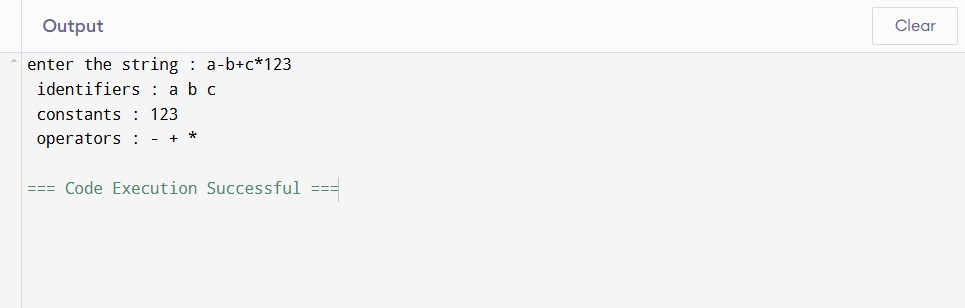
printf("\noperators:"); for(j=0;j<oc;j++)

{

printf("%c",operators[j]);

}

}



# Exp.No.2

**Develop alexical Analyzerto identify whether a given line is a comment or not using C**

**Program:** #include<stdio.h> #include<conio.h>int main()

{

charcom[30]; int i=2,a=0;

printf("\nEntercomment:"); gets(com);

if(com[0]=='/')

{

if(com[1]=='/')

printf("\nItisacomment"); else if(com[1]=='\*')

{

for(i=2;i<=30;i++)

{

if(com[i]=='\*'&&com[i+1]=='/')

{

}

else

}

printf("\nItisacomment"); a=1;

break;

continue;

}

else

}

else

if(a==0)

printf("\nItis notacomment");

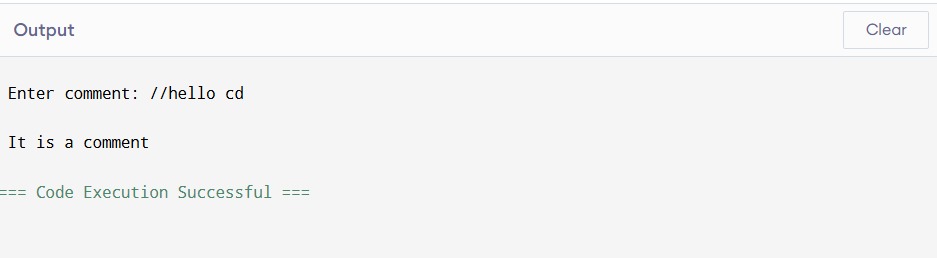
printf("\nItis notacomment");

printf("\nItis notacomment");

}

## Output:

**Input:**Entercomment://hello **Output**: It is a comment **Input:** Enter comment: hello **Output**: It is not a comment



# Exp.No.3

## DesignalexicalAnalyzerforgivenlanguageshouldignorethe redundant spaces, tabs and new lines and ignore comments using C

**Program:**

#include<stdio.h> #include<stdlib.h> #include<string.h> #include<ctype.h>

intisKeyword(charbuffer[]){ char keywords[32][10] =

{"main","auto","break","case","char","const","continue","default",

"do","double","else","enum","extern","float","for","goto",

"if","int","long","register","return","short","signed",

"sizeof","static","struct","switch","typedef",

"unsigned","void","printf","while"}; int i, flag = 0;

for(i=0;i<32;++i)

{

if(strcmp(keywords[i],buffer)==0)

{

flag=1; break;

}

}

returnflag;

}

intmain()

{

charch,buffer[15],operators[]="+-\*/%="; FILE \*fp;

inti,j=0;

fp=fopen("flex\_input.txt","r"); if(fp == NULL){

printf("errorwhileopeningthefile\n");

exit(0);

}

while((ch=fgetc(fp))!=EOF){ for(i = 0; i < 6; ++i){

if(ch == operators[i]) printf("%cisoperator\n",ch);

}

if(isalnum(ch)){ buffer[j++]=ch;

}

elseif((ch==''||ch=='\n')&&(j!=0)){ buffer[j] = '\0';

j=0;

if(isKeyword(buffer) == 1) printf("%siskeyword\n",buffer); else

printf("%sisidentifier\n",buffer);

}

}

fclose(fp); return 0;

}

**Input:**flex\_input.txt main ( )

{

inta,b,c; c = b + c;

printf("%d",c);

}

## Output:

mainiskeyword int is keyword

ais indentifier

bis indentifier

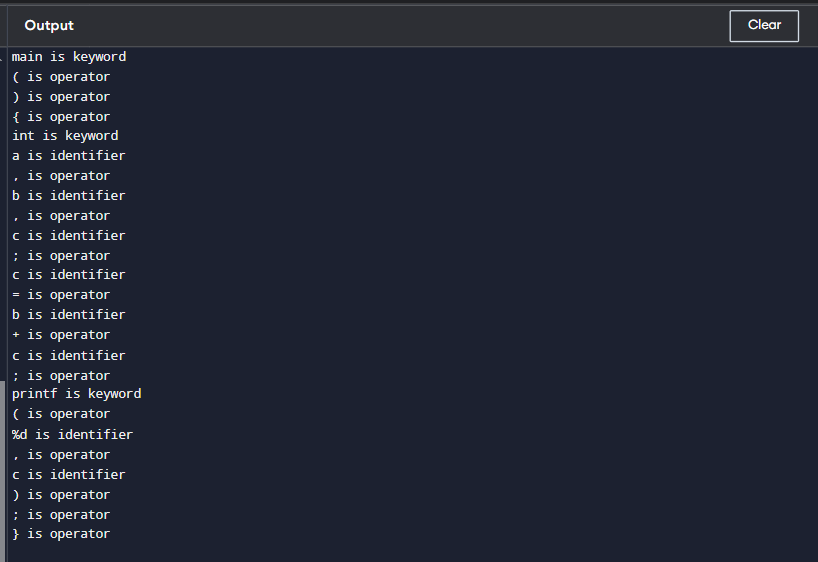
cisindentifier cis indentifier

= is operatorbisindentifier

+ is operatorcisindentifier

printfis keyword

% is operator disindentifier cis indentifier



# Exp.No.4

**Designalexical Analyzertovalidateoperatorstorecognizetheoperators+,-,\*,/ using regular arithmetic operators using C**

**Program:** #include<stdio.h> #include<conio.h>int main()

{

chars[5];

printf("\nEnteranyoperator:"); gets(s);

switch(s[0])

{

case'>':

if(s[1]=='=')

printf("\nGreaterthanorequal");

else

printf("\nGreaterthan");

break; case'<':

if(s[1]=='=')

printf("\nLessthanor equal");

else

printf("\nLessthan");

break; case'=':

if(s[1]=='=')

printf("\nEqualto");

else

printf("\nAssignment");

break; case'!':

if(s[1]=='=')

printf("\nNotEqual");

else

printf("\nBitNot");

break; case'&':

if(s[1]=='&')

printf("\nLogicalAND");

else

printf("\nBitwiseAND");

break; case'|':

if(s[1]=='|')

printf("\nLogicalOR");

else

printf("\nBitwiseOR");

break; case'+':

printf("\nAddition"); break;

case'-':

printf("\nSubstraction"); break;

case'\*':

printf("\nMultiplication"); break;

case'/':

printf("\nDivision"); break;

case'%':

printf("Modulus"); break;

default:

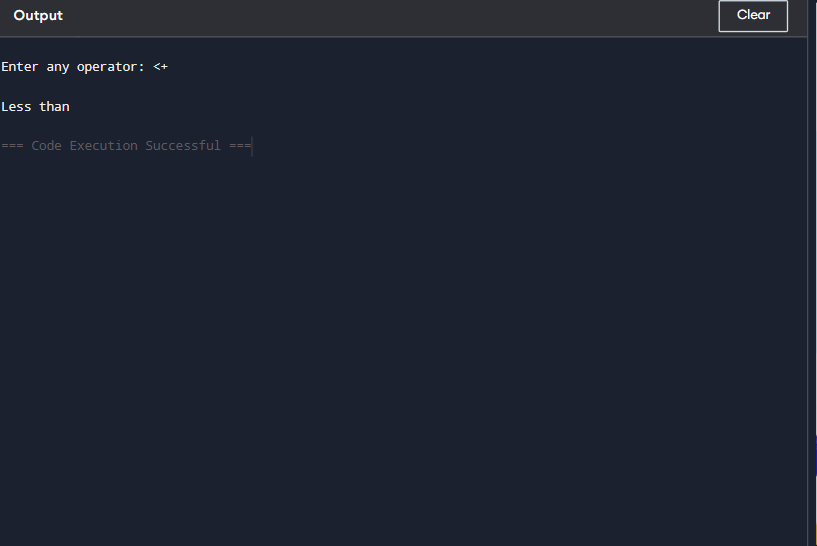
printf("\nNotaoperator");

}

}

## Output:

Enteranyoperator:<= Less than or equal



# Exp.No.5

**Designalexical Analyzertofindthenumberofwhitespacesandnewlinecharacters using C.**

## Program:

#include<stdio.h>

int main() {charstr[100];

intwords=0,lines=0,characters=0;

printf("Entertext(upto100characters,use~toend):\n"); scanf("%[^~]", str);

for(inti=0;str[i]!='\0';i++) {

if(str[i]==''||str[i]=='\t'){ words++;

}elseif(str[i]=='\n'){ lines++;

}else{

characters++;

}

}

//Checkforanemptyinput if (characters > 0) {

words++;//Iftherearecharacters,thereisatleastoneword lines++; // If there are characters, there is at least one line

}

printf("Total number of words: %d\n", words); printf("Total number of lines: %d\n", lines); printf("Totalnumberofcharacters:%d\n",characters);

return0;

}

## Output:

voidmain()

{

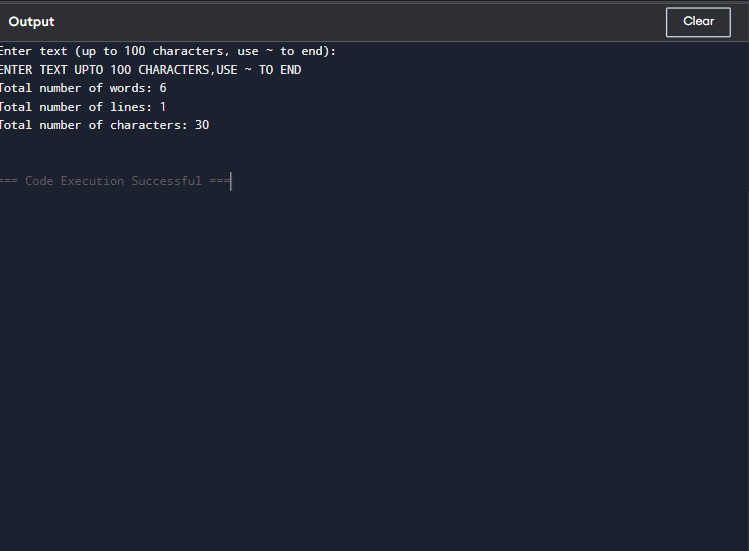
inta; intb;

a=b+c; c= d\*e;

}

Totalnumberofwords:12 Total number of lines : 7

TotalnumberofCharacter: 34



# Exp.No.6

**DevelopalexicalAnalyzertotestwhetheragivenidentifierisvalidornotusing C.**

## Program:

#include <stdio.h> #include<ctype.h>

int main() { chara[10];

intflag=1, i= 1;

printf("\nEnteranidentifier:"); fgets(a, sizeof(a), stdin);

if(isalpha(a[0])) {

while(a[i]!='\0'){

if(!isdigit(a[i])&&!isalpha(a[i])){ flag = 0;

break;

} i++;

}

}else{

flag=0;

}

if(flag==1){

printf("\nValididentifier\n");

}else{

printf("\nNotavalididentifier\n");

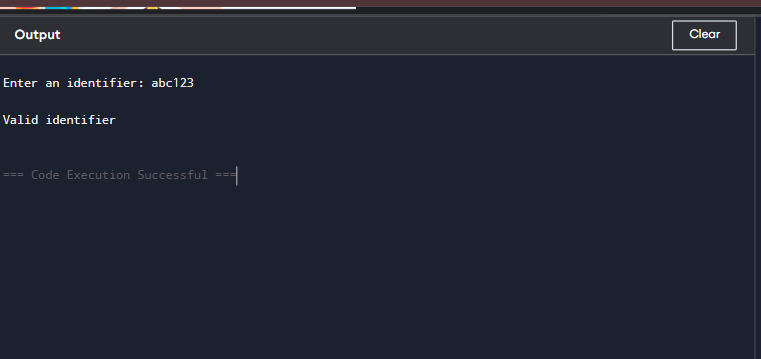
}

return0;

}

## Output:

Enteranidentifier:abc123 Valid identifier



# Exp.No.7

**WriteaCprogramtofindFIRST()-predictiveparserforthegiven grammar**

S→ AaAb/BbBa

A→∈

B→∈

**Program:** #include<stdio.h> #include<ctype.h>

voidFIRST(char[],char);

voidaddToResultSet(char[],char); int numOfProductions;

charproductionSet[10][10]; int main()

{

inti;

charchoice; char c;

charresult[20];

printf("Howmanynumberofproductions?:"); scanf(" %d",&numOfProductions);

for(i=0;i<numOfProductions;i++)//readproductionstringeg:E=E+T

{

printf("EnterproductionsNumber%d:",i+1); scanf(" %s",productionSet[i]);

}

do

{

printf("\nFindtheFIRSTof:"); scanf(" %c",&c);

FIRST(result,c);//ComputeFIRST;GetAnswerin'result'array printf("\n FIRST(%c)= { ",c);

for(i=0;result[i]!='\0';i++)

printf(" %c ",result[i]); //Displayresult printf("}\n");

printf("press'y'tocontinue:"); scanf(" %c",&choice);

}

while(choice=='y'||choice=='Y');

}

/\*

\*FunctionFIRST:

\*ComputetheelementsinFIRST(c)andwritethem

\*inResultArray.

\*/

voidFIRST(char\*Result,charc)

{

inti,j,k;

charsubResult[20]; int foundEpsilon; subResult[0]='\0'; Result[0]='\0';

//IfXisterminal,FIRST(X)={X}. if(!(isupper(c)))

{

addToResultSet(Result,c); return ;

}

//IfXisnonterminal

//Read each production for(i=0;i<numOfProductions;i++)

{

//FindproductionwithXasLHS if(productionSet[i][0]==c)

{

//If X → ε is a production, then add ε to FIRST(X). if(productionSet[i][2]=='$')addToResultSet(Result,'$');

//IfXisanon-terminal,andX→Y1Y2…Yk

//isaproduction,thenaddato FIRST(X)

//ifforsomei,aisinFIRST(Yi),

//andεisinallofFIRST(Y1),…,FIRST(Yi-1). else

{

j=2;

while(productionSet[i][j]!='\0')

{

foundEpsilon=0; FIRST(subResult,productionSet[i][j]); for(k=0;subResult[k]!='\0';k++)

addToResultSet(Result,subResult[k]); for(k=0;subResult[k]!='\0';k++)

if(subResult[k]=='$')

{

foundEpsilon=1; break;

}

//Noεfound,noneedtochecknextelement if(!foundEpsilon)

break; j++;

}

}

}

}

return;

}

/\*addToResultSetaddsthecomputed

\*elementtoresult set.

\*Thiscodeavoidsmultipleinclusionofelements

\*/

voidaddToResultSet(charResult[],charval)

{

intk;

for(k=0;Result[k]!='\0';k++) if(Result[k]==val)

return;

Result[k]=val;

Result[k+1]='\0';

}

## Output:

Howmanynumberofproductions?:4 Enter productions Number 1 : S=AaAb Enter productions Number 2 : S=BbBa Enter productions Number 3 : A=$ Enter productions Number 4 : B=$

FindtheFIRSTof:S FIRST(S)= {$ab }

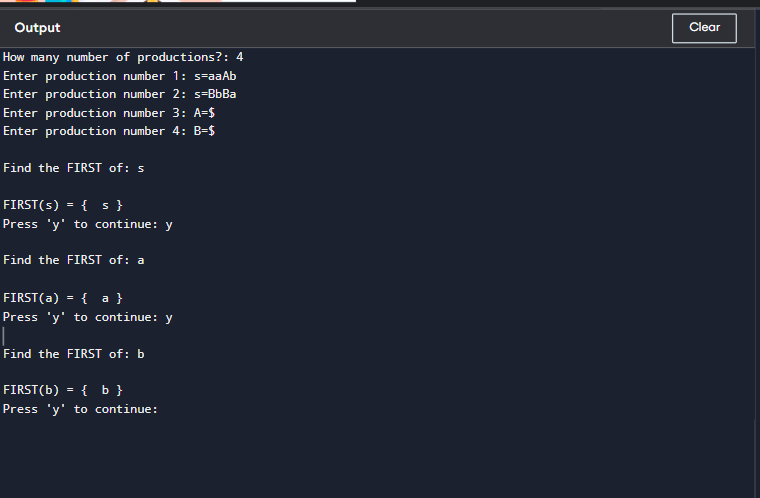
press'y'tocontinue:y

FindtheFIRSTof:A FIRST(A)= {$ }

press'y'tocontinue:y

FindtheFIRSTof:B FIRST(B)= {$ }

press'y'tocontinue:n



# Exp.No.8

**Writea C programtofindFOLLOW( )-predictiveparserfor thegiven grammar**

S→ AaAb/ BbBa

A→∈

B→∈

**Program:** #include<stdio.h> #include<ctype.h> #include<string.h>int limit, x = 0;

charproduction[10][10],array[10];

void find\_first(char ch); voidfind\_follow(charch);

voidArray\_Manipulation(charch);

intmain()

{

intcount;

charoption,ch;

printf("\nEnterTotalNumberofProductions:\t"); scanf("%d", &limit);

for(count=0;count<limit;count++)

{

printf("\nValueofProductionNumber[%d]:\t",count+1); scanf("%s", production[count]);

}

do

{

x =0;

printf("\nEnterproductionValuetoFindFollow:\t"); scanf(" %c", &ch);

find\_follow(ch);

printf("\nFollowValueof%c:\t{",ch); for(count = 0; count < x; count++)

{

printf("%c",array[count]);

}

printf("}\n");

printf("ToContinue,PressY:\t"); scanf(" %c", &option);

}while(option=='y'||option=='Y'); return 0;

}

voidfind\_follow(charch)

{

inti,j;

intlength=strlen(production[i]); if(production[0][0] == ch)

{

Array\_Manipulation('$');

}

for(i=0; i<limit; i++)

{

for(j=2;j <length;j++)

{

if(production[i][j]==ch)

{

if(production[i][j+1]!='\0')

{

find\_first(production[i][j+1]);

}

if(production[i][j+1]=='\0'&&ch!=production[i][0])

{

find\_follow(production[i][0]);

}

}

}

}

}

voidfind\_first(charch)

{

int i, k; if(!(isupper(ch)))

{

Array\_Manipulation(ch);

}

for(k=0;k <limit; k++)

{

if(production[k][0]==ch)

{

if(production[k][2]=='$')

{

find\_follow(production[i][0]);

}

else if(islower(production[k][2]))

{

Array\_Manipulation(production[k][2]);

}

else

{

find\_first(production[k][2]);

}

}

}

}

voidArray\_Manipulation(charch)

{

intcount;

for(count=0;count<=x; count++)

{

if(array[count]==ch)

{

return;

}

}

array[x++]= ch;

}

## Output:

Enter Total Number of Productions: 4 ValueofProductionNumber[1]:S=AaAb ValueofProductionNumber[2]:S=BbBa Value of Production Number [3]: A=$ Value of Production Number [4]: B=$ EnterproductionValuetoFindFollow:S

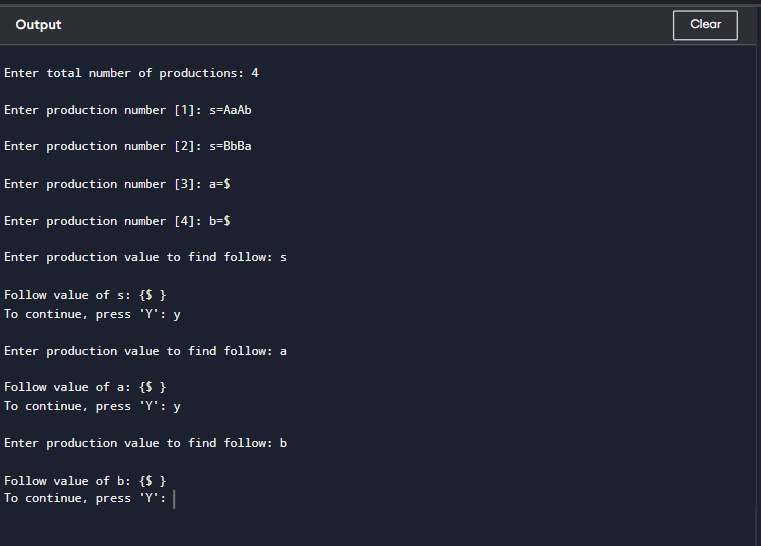
Follow Value of S: {$} To Continue, Press Y:y

EnterproductionValuetoFindFollow:A Follow Value of A: { a b }

ToContinue,PressY:y

EnterproductionValuetoFindFollow:B Follow Value of B: { b a }

ToContinue,PressY:n



# Exp.No.9

**ImplementaCprogramtoeliminateleftrecursionfromagivenCFG.**

S → (L) / aL→L,S/S

## Program:

#include<stdio.h> #include<string.h>#define SIZE 10int main () {

charnon\_terminal; char beta,alpha;int num;

charproduction[10][SIZE];

intindex=3;/\*startingofthestringfollowing"->"\*/ printf("Enter Number of Production : "); scanf("%d",&num);

printf("EnterthegrammarasE->E-A:\n"); for(int i=0;i<num;i++){

scanf("%s",production[i]);

}

for(inti=0;i<num;i++){

printf("\nGRAMMAR:::%s",production[i]); non\_terminal=production[i][0]; if(non\_terminal==production[i][index]) {

alpha=production[i][index+1]; printf(" is left recursive.\n");

while(production[i][index]!=0&&production[i][index]!='|') index++;

if(production[i][index]!=0) { beta=production[i][index+1]; printf("Grammarwithoutleftrecursion:\n");

printf("%c->%c%c\'",non\_terminal,beta,non\_terminal); printf("\n%c\'->%c%c\'|E\n",non\_terminal,alpha,non\_terminal);

}

else

printf("can'tbereduced\n");

}

else

printf("isnotleftrecursive.\n"); index=3;

}

}

## Output:

EnterNumberofProduction:2 Enter the grammar as E->E-A :

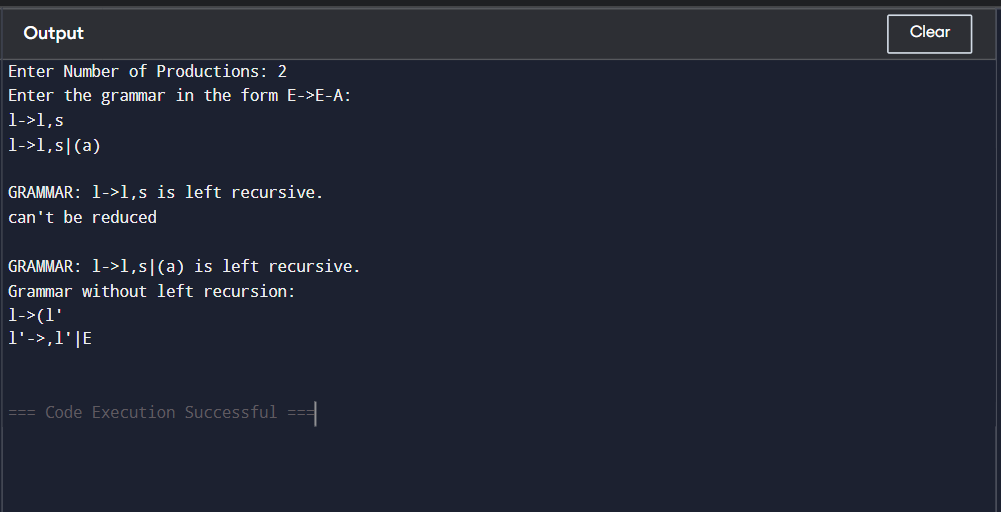
S->(L)|a L->L,S|S

GRAMMAR:::S->(L)|aisnotleftrecursive. GRAMMAR : : : L->L,S|S is left recursive.

Grammarwithoutleftrecursion:

L->SL'

L'->,L'|E



# Exp.No.10

## ImplementaCprogramtoeliminateleftfactoringfromagivenCFG.

S→iEtS/iEtSeS/a E → b

**Program:** #include<stdio.h> #include<string.h>

intmain()

{

chargram[20],part1[20],part2[20],modifiedGram[20],newGram[20]; int i, j = 0, k = 0, l = 0, pos;

// Input production printf("EnterProduction:S->"); gets(gram);

//Extractpart1andpart2

for(i=0;gram[i]!='|';i++,j++) part1[j] = gram[i];

part1[j]='\0';

for(j=++i,i=0;gram[j]!='\0';j++,i++) part2[i] = gram[j];

part2[i]='\0';

//Findcommonprefix

for(i=0;part1[i]==part2[i];i++)

{

modifiedGram[k]=part1[i]; k++;

pos=i +1;

}

//Createmodifiedproduction modifiedGram[k] = 'X';

modifiedGram[++k]='\0';

//Createnew production

for(i=pos,j=0;part1[i]!='\0';i++,j++) newGram[j] = part1[i];

newGram[j++]='|';

for(i=pos;part2[i]!='\0';i++,j++) newGram[j] = part2[i];

newGram[j]='\0';

//Printtheresult

printf("\nS->%s",modifiedGram); printf("\n X->%s\n", newGram);

return0;

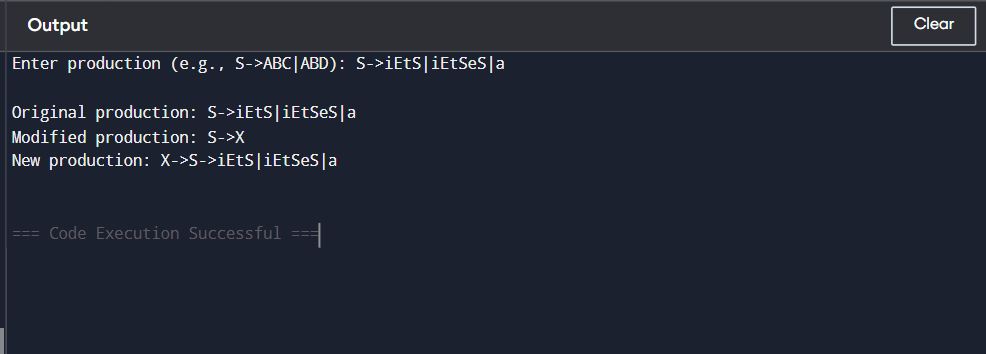
}

## Output:

EnterProduction:S->iEtS|iEtSeS|a

S->iEtSX

X->|eS|a



# Exp.No.11

## ImplementaCprogramtoperformsymboltableoperations.

**Program:** #include<stdio.h> #include<stdlib.h> #include<string.h>int cnt=0;

structsymtab

{

charlabel[20]; int addr;

}

sy[50];

voidinsert();

intsearch(char\*); void display(); void modify();

intmain()

{

int ch,val; charlab[10]; do

{

printf("\n1.insert\n2.display\n3.search\n4.modify\n5.exit\n"); scanf("%d",&ch);

switch(ch)

{

case 1:

insert(); break; case 2:

display(); break;

case 3: printf("enterthelabel");

scanf("%s",lab); val=search(lab);

if(val==1)

printf("labelisfound"); else

printf("labelisnotfound"); break;

case 4:

modify(); break;

case 5:

exit(0); break;

}

}while(ch<5);

}

voidinsert()

{

intval;

charlab[10]; int symbol;

printf("enterthelabel"); scanf("%s",lab); val=search(lab); if(val==1)

printf("duplicatesymbol"); else

{

strcpy(sy[cnt].label,lab); printf("entertheaddress"); scanf("%d",&sy[cnt].addr); cnt++;

}

}

intsearch(char\*s)

{

intflag=0,i;for(i=0;i<cnt;i++)

{

if(strcmp(sy[i].label,s)==0)

flag=1;

}

returnflag;

}

voidmodify()

{

int val,ad,i; charlab[10];

printf("enterthelabe:"); scanf("%s",lab); val=search(lab); if(val==0)

printf("nosuchsymbol"); else

{

printf("label is found \n"); printf("entertheaddress"); scanf("%d",&ad); for(i=0;i<cnt;i++)

{

if(strcmp(sy[i].label,lab)==0) sy[i].addr=ad;

}

}

}

voiddisplay()

{

int i; for(i=0;i<cnt;i++)

printf("%s\t%d\n",sy[i].label,sy[i].addr);

}

## Output:

1.insert 2.display

3.search 4.modify 5.exit

1

enter the label a entertheaddress100

1.insert 2.display 3.search 4.modify 5.exit

2

a 100

1.insert 2.display 3.search 4.modify 5.exit

3

enterthelabela label is found 1.insert2.display 3.search 4.modify

5.exit 4

enterthelabe:a label is found

entertheaddress 200

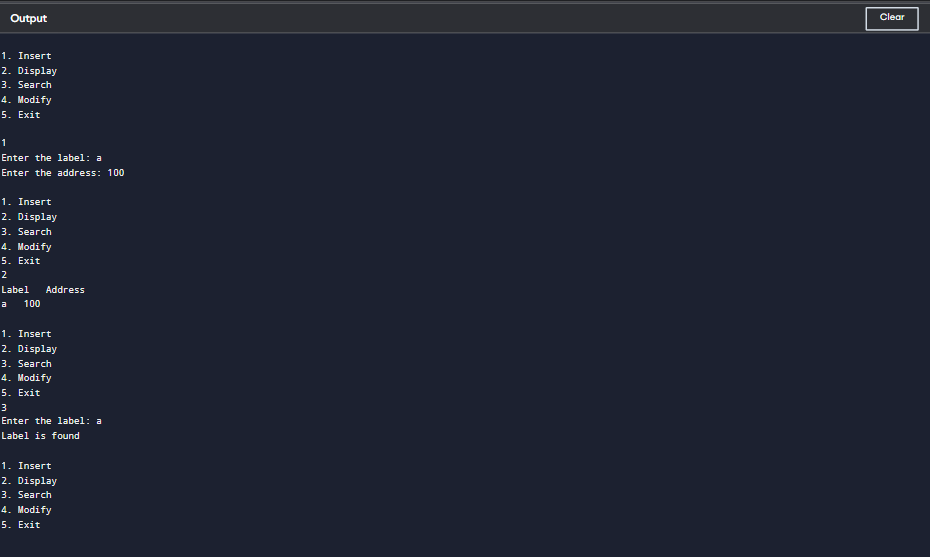
1.insert 2.display 3.search 4.modify

5.exit 2

a 200

1.insert 2.display 3.search 4.modify 5.exit

5



# Exp.No.12

## WriteaCprogramtoconstructrecursivedescentparsingforthegiven grammar

E → TE’

E’→ +TE’/ ∈

T → FT’

T’→\*FT’/∈

F→ (E )/id

## Program:

#include<stdio.h> #include<conio.h> #include<string.h>char input[100]; int i,l;

voidmain()

{

//clrscr();

printf("\nRecursivedescentparsingforthefollowinggrammar\n");printf("\nE-

>TE'\nE'->+TE'/@\nT->FT'\nT'->\*FT'/@\nF->(E)/ID\n");printf("\nEnterthe string to be checked:"); gets(input);

if(E())

{

if(input[i+1]=='\0') printf("\nStringisaccepted"); else

printf("\nStringisnotaccepted");

}

else

printf("\nStringnotaccepted"); getch();

} E()

{

if(T())

{ if(EP())

return(1); else return(0);

}

else return(0);

} EP()

{

if(input[i]=='+')

{ i++;

if(T())

{ if(EP())

return(1); else return(0);

}

else return(0);

}

else return(1);

} T()

{ if(F())

{ if(TP())

return(1); else return(0);

}

else return(0);

} TP()

{

if(input[i]=='\*')

{ i++;

if(F())

{ if(TP())

return(1); else return(0);

}

else return(0);

}

else return(1);

} F()

{

if(input[i]=='(')

{ i++;

if(E())

{

if(input[i]==')')

{ i++;

return(1);

}

else return(0);

}

else

return(0);

}

elseif(input[i]>='a'&&input[i]<='z'||input[i]>='A'&&input[i]<='Z')

{ i++;

return(1);

}

else return(0);

}

## Output:

Recursivedescentparsingforthefollowinggrammar

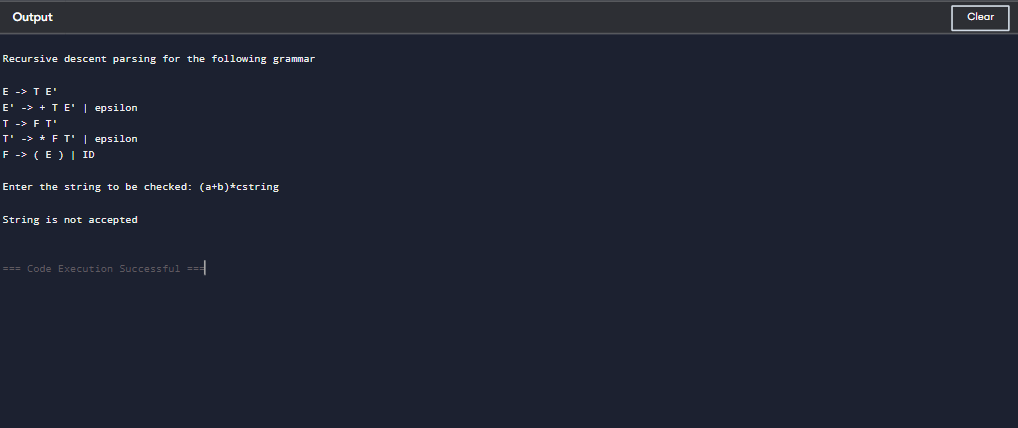
E->TE'

E'->+TE'/@ T->FT'

T'->\*FT'/@ F->(E)/ID

Enterthestringtobechecked:(a+b)\*c String is accepted

Enterthestringtobechecked:a/c+d String is not accepted



# Exp.No.13

## WriteaCprogramtoimplementeitherTopDownparsingtechniqueor Bottom Up Parsingtechnique to check whether the given input string is satisfying the grammar or not.

**Program:**

#include<stdio.h> #include<conio.h> #include<string.h>int main() {

char string[50]; intflag,count=0;

printf("Thegrammaris:S->aS,S->Sb,S->ab\n"); printf("Enter the string to be checked:\n"); gets(string);

if(string[0]=='a'){

flag=0;

for(count=1;string[count-1]!='\0';count++){ if(string[count]=='b') {

flag=1; continue;

}elseif((flag==1)&&(string[count]=='a')){

printf("Thestringdoesnotbelongtothespecified

grammar");

break;

}elseif(string[count]=='a')

continue;elseif((flag==1)&&(string[count]='\0')){ printf("String not accepted…..!!!!");

break;

}else{

printf("Stringaccepted");

}

}

}

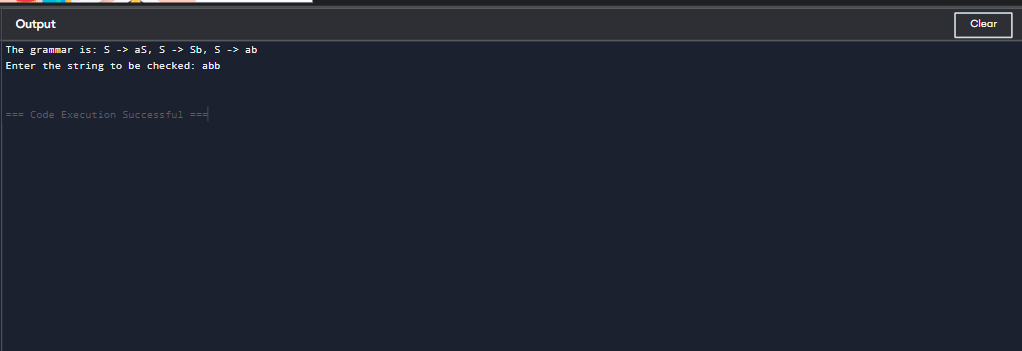
}

## Output:

Thegrammaris:S->aS,S->Sb,S->ab

Enterthestringtobechecked: abb

Stringaccepted



# Exp.No.14

## ImplementtheconceptofShiftreduceparsinginCProgramming.

**Program:**

#include<stdio.h> #include<stdlib.h> #include<conio.h> #include<string.h>

charip\_sym[15],stack[15];intip\_ptr=0,st\_ptr=0,len,i;chartemp[2],temp2[2]; char act[15];

voidcheck();intmain()

{

//clrscr();

printf("\n\t\tSHIFTREDUCEPARSER\n");printf("\nGRAMMER\n");

printf("\nE->E+E\nE->E/E");printf("\nE->E\*E\nE->a/b");printf("\nenterthe input symbol:\t"); gets(ip\_sym);

printf("\n\tstackimplementationtable");printf("\nstack\t\tinputsymbol\t\t action");

printf("\n \t\t \t\t \n");

printf("\n$\t\t%s$\t\t\t--",ip\_sym);strcpy(act,"shift"); temp[0]=ip\_sym[ip\_ptr]; temp[1]='\0';

strcat(act,temp);len=strlen(ip\_sym);for(i=0;i<=len-1;i++)

{

stack[st\_ptr]=ip\_sym[ip\_ptr];

stack[st\_ptr+1]='\0';ip\_sym[ip\_ptr]='';ip\_ptr++;

printf("\n$%s\t\t%s$\t\t\t%s",stack,ip\_sym,act);strcpy(act,"shift"); temp[0]=ip\_sym[ip\_ptr]; temp[1]='\0'; strcat(act,temp); check(); st\_ptr++;

}

st\_ptr++;check();

}

voidcheck()

{

intflag=0;temp2[0]=stack[st\_ptr];temp2[1]='\0'; if((!strcmpi(temp2,"a"))||(!strcmpi(temp2,"b")))

{

stack[st\_ptr]='E';if(!strcmpi(temp2,"a"))

printf("\n $%s\t\t%s$\t\t\tE->a",stack,ip\_sym); else printf("\n$%s\t\t%s$\t\t\tE->b",stack,ip\_sym);flag=1;

}

if((!strcmpi(temp2,"+"))||(strcmpi(temp2,"\*"))||(!strcmpi(temp2,"/")))

{

flag=1;

}

if((!strcmpi(stack,"E+E"))||(!strcmpi(stack,"E\E"))||(!strcmpi(stack,"E\*E")))

{

strcpy(stack,"E"); st\_ptr=0; if(!strcmpi(stack,"E+E")) printf("\n$%s\t\t%s$\t\t\tE->E+E",stack,ip\_sym);else if(!strcmpi(stack,"E\E"))

printf("\n$%s\t\t%s$\t\t\tE->E\E",stack,ip\_sym);else if(!strcmpi(stack,"E\*E"))

printf("\n $%s\t\t%s$\t\t\tE->E\*E",stack,ip\_sym); else printf("\n$%s\t\t%s$\t\t\tE->E+E",stack,ip\_sym);flag=1;

}

if(!strcmpi(stack,"E")&&ip\_ptr==len)

{

printf("\n$%s\t\t%s$\t\t\tACCEPT",stack,ip\_sym);getch(); exit(0);

}

if(flag==0)

{

printf("\n%s\t\t\t%s\t\treject",stack,ip\_sym);exit(0);

}

return;

}

## Output:

SHIFTREDUCEPARSER GRAMMER

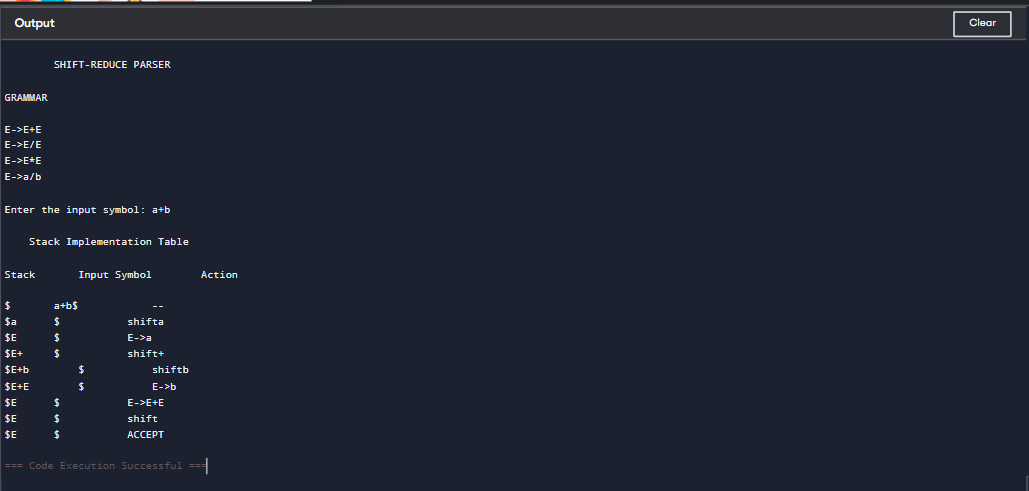
E->E+E E->E/E E->E\*E

E->a/b

entertheinputsymbol: a+b

stackimplementationtable

|  |  |  |  |
| --- | --- | --- | --- |
| stack  $ | inputsymbol  a+b$ | -- | action |
| $a | +b$ | shifta | |
| $E | +b$ | E->a | |
| $E+ | b$ | shift+ | |
| $E+b | $ | shiftb | |
| $E+E | $ | E->b | |
| $E | $ | E->E+E | |
| $E | $ | ACCEPT | |
|  |  |  | |
|  |  |  | |



# Exp.No.15

## WriteaCProgramtoimplementtheoperatorprecedenceparsing.

**Program:**

#include<stdio.h> #include<string.h>

char\*input; int i=0;

charlasthandle[6],stack[50],handles[][5]={")E(","E\*E","E+E","i","E^E"};

//(E)becomes)E(whenpushedtostack

inttop=0,l;

charprec[9][9]={

/\*input\*/

/\*stack + -\*/^i()$\*/

/\*+\*/'>','>','<','<','<','<','<','>','>',

/\*-\*/'>','>','<','<','<','<','<','>','>',

/\*\*\*/'>','>','>','>','<','<','<','>','>',

/\*/\*/'>','>','>','>','<','<','<','>','>',

/\*^\*/'>','>','>','>','<','<','<','>','>',

/\*i\*/'>','>','>','>','>','e','e','>','>',

/\*(\*/'<','<','<','<','<','<','<','>','e',

/\*)\*/'>','>','>','>','>','e','e','>','>',

/\*$\*/'<','<','<','<','<','<','<','<','>',

};

intgetindex(charc)

{

switch(c)

{

case'+':return0;

case'-':return1;

case'\*':return2;

case'/':return3;

case'^':return4;

case'i':return5;

case'(':return6;

case')':return7;

case'$':return8;

}

}

intshift()

{

stack[++top]=\*(input+i++); stack[top+1]='\0';

}

intreduce()

{

int i,len,found,t; for(i=0;i<5;i++)//selectinghandles

{

len=strlen(handles[i]); if(stack[top]==handles[i][0]&&top+1>=len)

{

found=1; for(t=0;t<len;t++)

{

if(stack[top-t]!=handles[i][t])

{

found=0; break;

}

}

if(found==1)

{

stack[top-t+1]='E'; top=top-t+1;

strcpy(lasthandle,handles[i]); stack[top+1]='\0';

return1;//successfulreduction

}

}

}

return0;

}

voiddispstack()

{

int j; for(j=0;j<=top;j++)

printf("%c",stack[j]);

}

voiddispinput()

{

int j; for(j=i;j<l;j++)

printf("%c",\*(input+j));

}

voidmain()

{

intj;

input=(char\*)malloc(50\*sizeof(char)); printf("\nEnter the string\n"); scanf("%s",input); input=strcat(input,"$");l=strlen(input);

strcpy(stack,"$"); printf("\nSTACK\tINPUT\tACTION"); while(i<=l)

{

shift(); printf("\n"); dispstack(); printf("\t"); dispinput(); printf("\tShift");

if(prec[getindex(stack[top])][getindex(input[i])]=='>')

{

while(reduce())

{

printf("\n"); dispstack(); printf("\t"); dispinput();

printf("\tReduced:E->%s",lasthandle);

}

}

}

if(strcmp(stack,"$E$")==0) printf("\nAccepted;");

else

printf("\nNotAccepted;");

}

## Output:

Enterthestring i\*(i+i)\*i

STACKINPUTACTION

$i \*(i+i)\*i$ Shift

$E \*(i+i)\*i$ Reduced:E->i

$E\* (i+i)\*i$ Shift

$E\*( i+i)\*i$ Shift

$E\*(i+i)\*i$ Shift

$E\*(E+i)\*i$ Reduced:E->i

$E\*(E+i)\*i$ Shift

$E\*(E+i)\*i$ Shift

$E\*(E+E)\*i$ Reduced:E->i

$E\*(E)\*i$ Reduced:E->E+E

$E\*(E)\*i$ Shift

$E\*E \*i$ Reduced:E->)E(

$E \*i$ Reduced:E->E\*E

$E\* i$ Shift

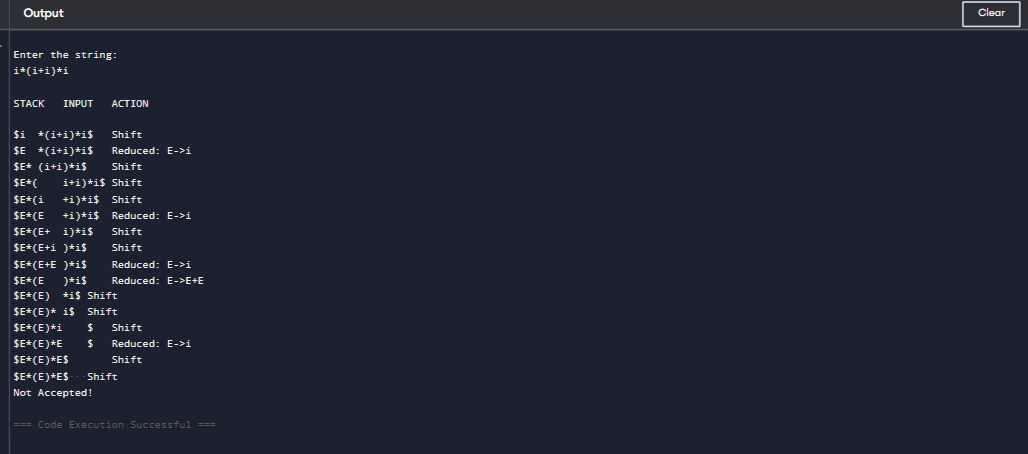
$E\*i $ Shift

$E\*E $ Reduced:E->i

$E $ Reduced:E->E\*E

$E$ Shift

$E$ Shift Accepted;



# Exp.No.16

## Writea CProgramto GeneratetheThreeaddress code representation for the given input statement.

**Program:**

#include<stdio.h> #include<conio.h> #include<stdlib.h> #include<string.h>struct three

{

chardata[10],temp[7];

}s[30];

intmain()

{

chard1[7],d2[7]="t"; int i=0,j=1,len=0; FILE \*f1,\*f2;

//clrscr(); f1=fopen("sum.txt","r");

f2=fopen("out.txt","w");

while(fscanf(f1,"%s",s[len].data)!=EOF) len++;

itoa(j,d1,7); strcat(d2,d1); strcpy(s[j].temp,d2);

strcpy(d1,"");

strcpy(d2,"t"); if(!strcmp(s[3].data,"+"))

{

fprintf(f2,"%s=%s+%s",s[j].temp,s[i+2].data,s[i+4].data); j++;

}

elseif(!strcmp(s[3].data,"-"))

{

fprintf(f2,"%s=%s-%s",s[j].temp,s[i+2].data,s[i+4].data); j++;

}

for(i=4;i<len-2;i+=2)

{

itoa(j,d1,7); strcat(d2,d1); strcpy(s[j].temp,d2);

if(!strcmp(s[i+1].data,"+")) fprintf(f2,"\n%s=%s+%s",s[j].temp,s[j-1].temp,s[i+2].data); else if(!strcmp(s[i+1].data,"-"))

fprintf(f2,"\n%s=%s-%s",s[j].temp,s[j-1].temp,s[i+2].data); strcpy(d1,"");

strcpy(d2,"t"); j++;

}

fprintf(f2,"\n%s=%s",s[0].data,s[j-1].temp); fclose(f1);

fclose(f2); getch();

}

## Output:

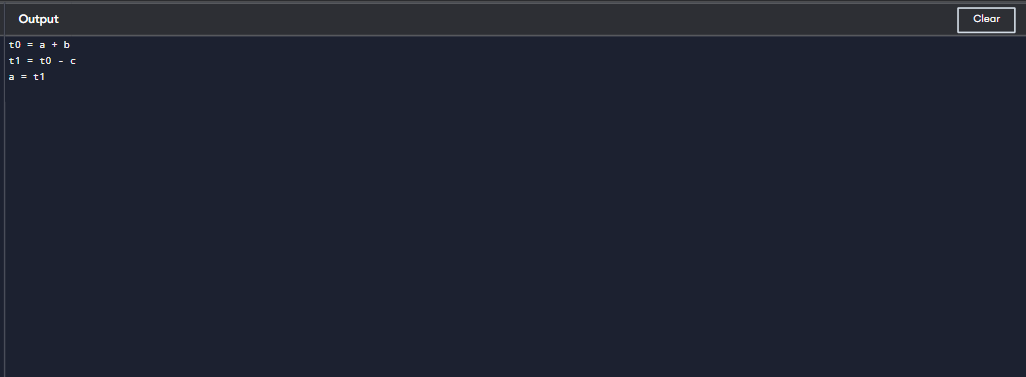
**Input:**sum.txt

out=in1+ in2+in3 -in4

**Output:**out.txt t1=in1+in2 t2=t1+in3

t3=t2-in4

out=t3



# Exp.No.17

## WriteaCprogramforimplementingaLexicalAnalyzertoScanand Count the number of characters, words, and lines in a file.

**Program:**

#include<stdio.h> int main()

{

charstr[100];//inputstringwithsize 100

intwords=0,newline=0,characters=0;//countervariables scanf("%[^~]",&str);//scanf formatting

for(inti=0;str[i]!='\0';i++)

{

if(str[i]=='')

{

words++;

}

elseif(str[i]=='\n')

{

newline++;

words++;//sincewitheverynextlinenewwordsstart.cornercase1

}

elseif(str[i]!=''&&str[i]!='\n'){ characters++;

}

}

if(characters>0)//Cornercase2,3.

{

words++; newline++;

}

printf("Total number of words : %d\n",words); printf("Total number of lines : %d\n",newline); printf("Total number of characters : %d\n",characters); return 0;

}

## Output:

voidmain()

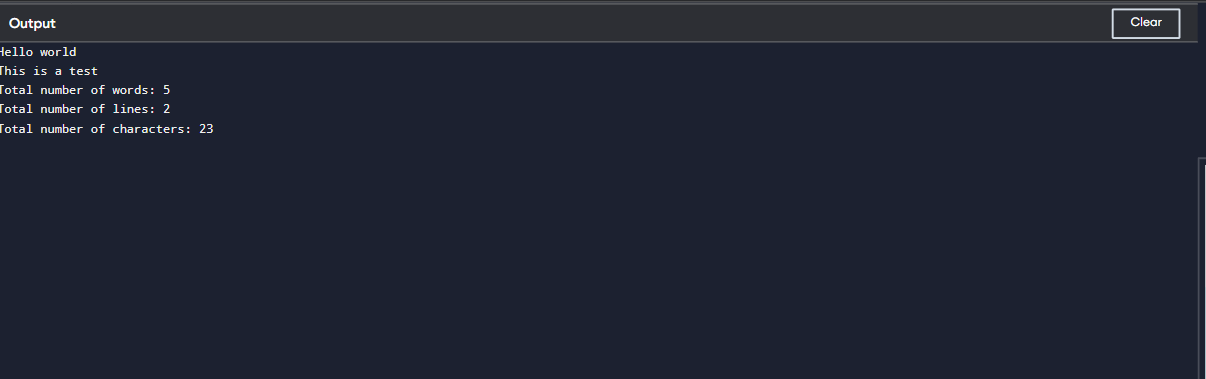
{

inta; intb;

a=b+c; c= d\*e;

}~

Totalnumberofwords:18 Total number of lines : 7



# Exp.No.18

## WriteaCprogramtoimplementthebackendofthecompiler.

**Program:** #include<stdio.h> #include<conio.h> #include<string.h>int main()

{

intn,i,j;

chara[50][50];

printf("entertheno:intermediatecode:"); scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("enterthe3addresscode:%d:",i+1); for(j=0;j<6;j++)

{

scanf("%c",&a[i][j]);

}

}

printf("thegeneratedcodeis:"); for(i=0;i<n;i++)

{

printf("\nmov%c,R%d",a[i][3],i);

if(a[i][4]=='-')

{

printf("\nsub%c,R%d",a[i][5],i);

}

if(a[i][4]=='+')

{

printf("\nadd%c,R%d",a[i][5],i);

}

if(a[i][4]=='\*')

{

printf("\nmul%c,R%d",a[i][5],i);

}

if(a[i][4]=='/')

{

printf("\ndiv%c,R%d",a[i][5],i);

}

printf("\nmovR%d,%c",i,a[i][1]); printf("\n");

}

return0;

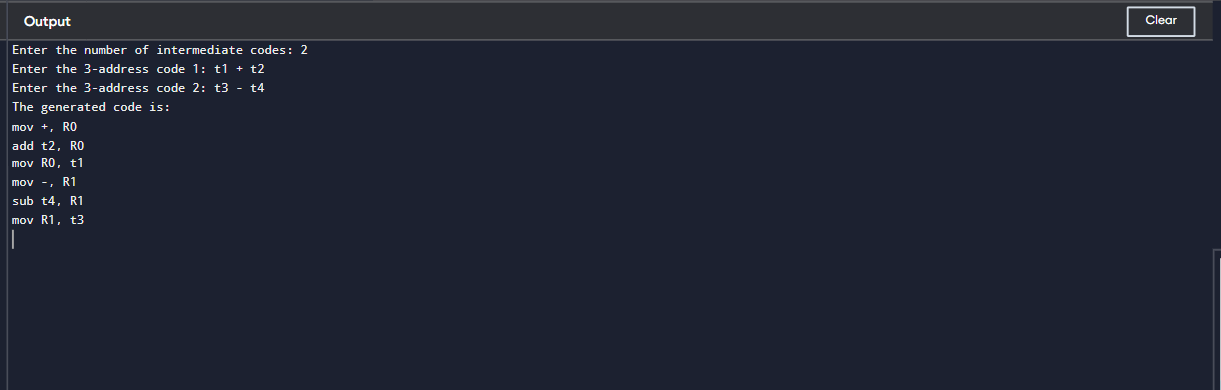
}

## Output:

entertheno:intermediatecode:2 enter the 3 address code:1:a=b+c enterthe3addresscode:2:d=n\*d the generated code is:

mova,R0 add c,R0 movR0,a

movn,R1 mul d,R1 movR1,d



# Exp.No.19

## WriteaCprogramtocomputeLEADING()–operatorprecedenceparser for the given grammar

E→E+T|T

T→T\*F|F F→(E)|id

## Program:

#include<conio.h> #include<stdio.h>

chararr[18][3]={{'E','+','F'},{'E','\*','F'},{'E','(','F'},{'E',')','F'},{'E','i','F'},{'E','$','F'},

{'F','+','F'},{'F','\*','F'},{'F','(','F'},{'F',')','F'},{'F','i','F'},{'F','$','F'},{'T','+','F'},

{'T','\*','F'},{'T','(','F'},{'T',')','F'},{'T','i','F'},{'T','$','F'}};

charprod[]="EETTFF";

charres[6][3]={{'E','+','T'},{'T','\0'},{'T','\*','F'},{'F','\0'},{'(','E',')'},{'i','\0'}};

charstack[5][2]; int top = -1;

voidinstall(charpro,charre){ int i;

for(i =0;i<18;++i){

if(arr[i][0]==pro&&arr[i][1]==re){

arr[i][2]='T';

break;

}

}

++top; stack[top][0]=pro; stack[top][1] = re;

}

int main() { inti=0,j;

charpro,re,pri='';

for(i =0;i<6;++i){

for(j=0;j<3&&res[i][j]!='\0';++j){

if(res[i][j]=='+'||res[i][j]=='\*'||res[i][j]=='('||res[i][j]==')'||res[i][j]== 'i' || res[i][j] == '$') {

install(prod[i],res[i][j]); break;

}

}

}

while(top>=0){

pro=stack[top][0]; re = stack[top][1];

--top;

for(i =0;i<6;++i){

if(res[i][0]==pro&&res[i][0]!=prod[i]){ install(prod[i], re);

}

}

}

for(i=0;i<18;++i){ printf("\n\t");

for (j = 0; j < 3; ++j) printf("%c\t",arr[i][j]);

}

getch(); printf("\n\n");

for(i=0;i<18;++i){ if (pri != arr[i][0]) {pri = arr[i][0];

printf("\n\t%c->",pri);

}

if(arr[i][2]=='T')

printf("%c",arr[i][1]);

}

getch();

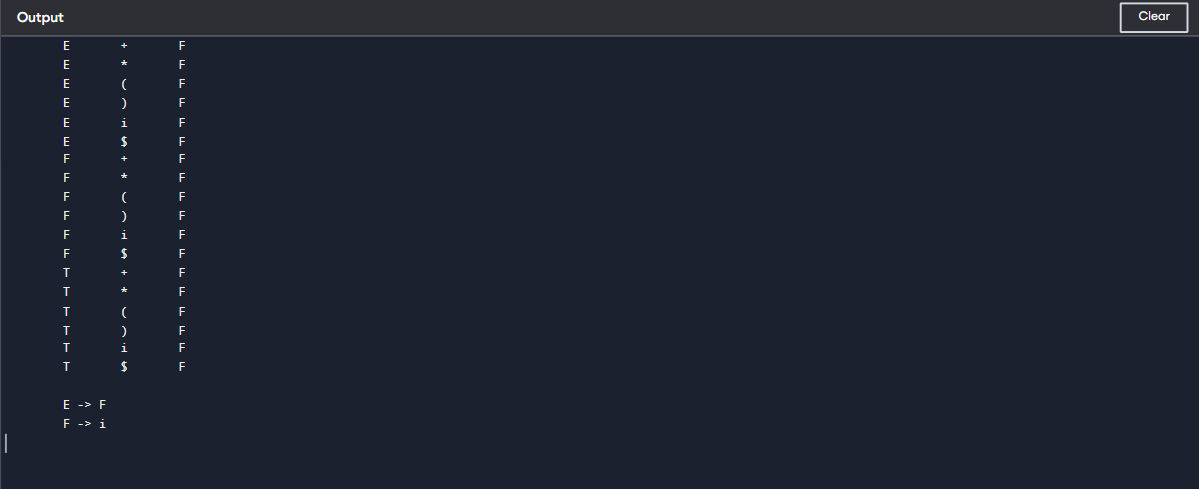
}

|  |  |  |
| --- | --- | --- |
| **Output:** |  | |
| E | + | T |
| E | \* | T |
| E | ( | T |
| E | ) | F |
| E | i | T |
| E | $ | F |
| F | + | F |
| F | \* | F |
| F | ( | T |
| F | ) | F |
| F | i | T |
| F | $ | F |
| T | + | F |
| T | \* | T |
| T | ( | T |
| T | ) | F |
| T | i | T |
| T | $ | F |

E->+\*(i

F->(i

T->\*(i



# Exp.No.20

**WriteaCprogramtocomputeTRAILING()–operatorprecedence parser for the given grammar**

**E →E +T |T**

**T→T\*F|F F→(E )| id**

**Program:**

#include<conio.h> #include<stdio.h>

chararr[18][3]={{'E','+','F'},{'E','\*','F'},{'E','(','F'}, {'E',')','F'}, {'E','i','F'},

{'E','$','F'}, {'F','+','F'}, {'F','\*','F'}, {'F','(','F'}, {'F',')','F'}, {'F','i','F'},

{'F','$','F'}, {'T','+','F'}, {'T','\*','F'}, {'T','(','F'}, {'T',')','F'}, {'T','i','F'},

{'T','$','F'},

};

charprod[6]="EETTFF";

char res[6][3] ={ {'E', '+', 'T'}, {'T', '\0', '\0'}, {'T', '\*', 'F'}, {'F', '\0', '\0'}, {'(','E', ')'},{'i', '\0', '\0'},};

charstack[5][2]; int top = -1;

voidinstall(charpro,charre){ int i;

for(i =0;i<18;++i){

if(arr[i][0]==pro&&arr[i][1]==re){

++top; arr[i][2]='T';

stack[top][0]=pro; stack[top][1] = re;

break;//Addedbreaktoexittheloopwhenthematchisfound

}

}

}

int main() { inti=0,j;

charpro,re,pri='';

for (i = 0; i < 6; ++i) { for(j=2;j>=0;--j){

if(res[i][j]=='+'||res[i][j]=='\*'||res[i][j]=='('||res[i][j]==')'||res[i][j]== 'i' || res[i][j] == '$') {

install(prod[i],res[i][j]); break;

}elseif(res[i][j]=='E'||res[i][j]=='F'||res[i][j]=='T'){

if(res[i][j-1]=='+'||res[i][j-1]=='\*'||res[i][j-1]=='('||res[i][j -

1]==')'||res[i][j -1]=='i'||res[i][j-1]=='$'){ install(prod[i], res[i][j - 1]);

break;

}

}

}

}

while(top>=0){

pro=stack[top][0]; re = stack[top][1];

--top;

for (i = 0; i < 6; ++i) { for(j=2;j>=0; --j){

if(res[i][0]==pro&&res[i][0]!=prod[i]){ install(prod[i], re);

break;

}elseif(res[i][0]!='\0')break;

}

}

}

for(i=0;i<18;++i){ printf("\n\t");

for (j = 0; j < 3; ++j) printf("%c\t",arr[i][j]);

}

printf("\n\n");

for(i=0;i<18;++i){ if (pri != arr[i][0]) {

pri=arr[i][0];

printf("\n\t%c->",pri);

}

if(arr[i][2]=='T')

printf("%c",arr[i][1]);}

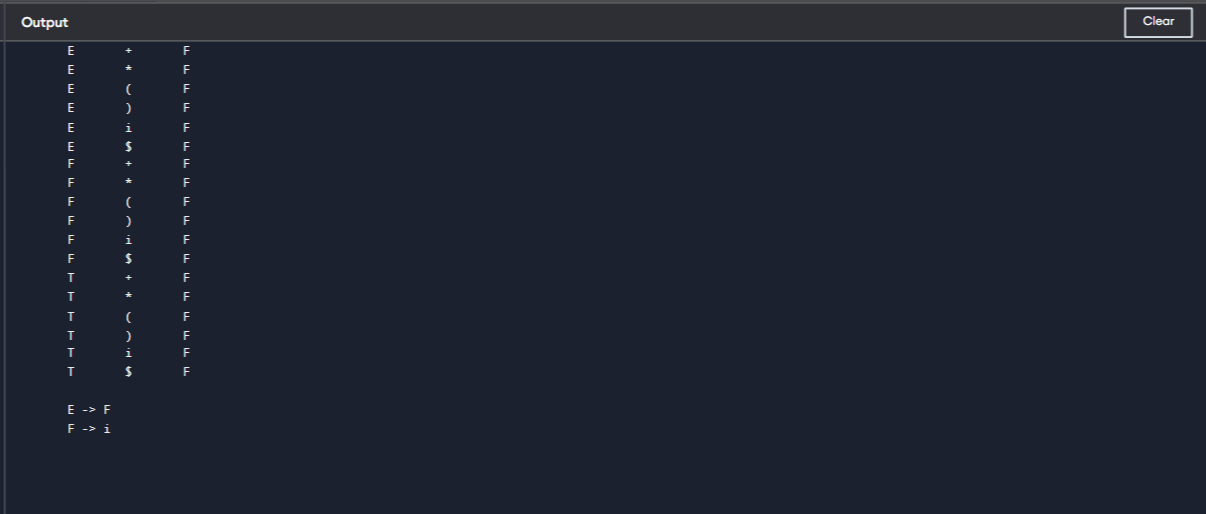
}

|  |  |  |
| --- | --- | --- |
| **Output:** |  | |
| E | + | F |
| E | \* | F |
| E | ( | F |
| E | ) | F |
| E | i | F |
| E | $ | F |
| F | + | F |
| F | \* | F |
| F | ( | F |
| F | ) | F |
| F | i | F |
| F | $ | F |
| T | + | F |
| T | \* | F |
| T | ( | F |
| T | ) | F |
| T | i | F |
| T | $ | F |

E->

F->

T->



# Exp.No.21

## Write a LEXspecification file to take input Cprogramfroma .c file and count tthe number ofcharacters, number of lines & number ofwords.

**InputSourceProgram:(sample.c)**

#include<stdio.h> int main()

{

int number1, number2, sum; printf("Entertwointegers:");

scanf("%d%d",&number1,&number2); sum = number1 + number2;

printf("%d+%d=%d",number1,number2,sum); return 0;

}

**Program:(count\_lines.l)**

%{

intnchar,nword,nline;

%}

%%

\n{nline++;nchar++;}

[^\t\n]+{nword++,nchar+=yyleng;}

.{nchar++;}

%%

intyywrap(void){ return 1;

}

intmain(intargc,char\*argv[]){ yyin = fopen(argv[1], "r"); yylex();

printf("Numberofcharacters=%d\n",nchar); printf("Number of words = %d\n", nword); printf("Number of lines = %d\n", nline); fclose(yyin);

}

## Output:

# Exp.No.22

## WriteaLEXprogram toprintall theconstantsinthegivenCsource program file.

**InputSourceProgram:(sample.c)**

#define P 314 #include<stdio.h> #include<conio.h>

void main()

{

int a,b,c = 30; printf("hello");

}

**Program:(countconstants.l)**

digit[0-9]

%{

intcons=0;

%}

%%

{digit}+{cons++;printf("%sisaconstant\n",yytext);}

.|\n{}

%%

intyywrap(void){ return 1; }

intmain(void)

{

FILE\*f;

charfile[10];

printf("EnterFileName:"); scanf("%s",file);

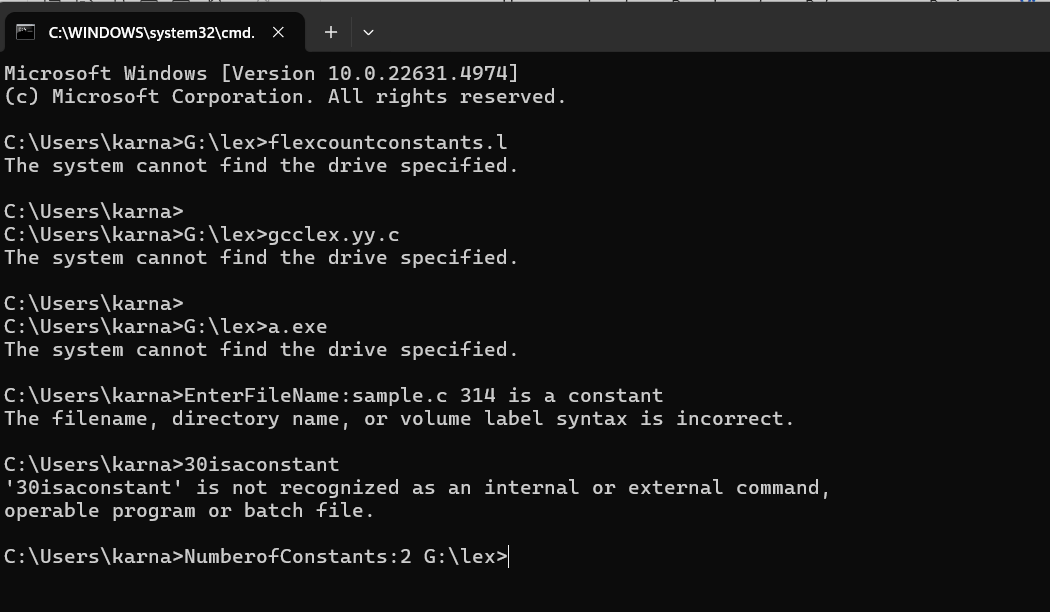
f=fopen(file,"r"); yyin = f;

yylex();

printf("NumberofConstants:%d\n",cons); fclose(yyin);

}

## Output:



# Exp.No.23

## WriteaLEXprogramtocountthenumberofMacrosdefinedandheader filesincluded in the C program.

**InputSourceProgram:(sample.c)**

#define PI 3.14 #include<stdio.h> #include<conio.h>void main()

{

int a,b,c = 30; printf("hello");

}

## Program:(count\_macro.l)

%{

intnmacro,nheader;

%}

%%

^#define{nmacro++;}

^#include{nheader++;}

.|\n{}

%%

intyywrap(void){ return 1;

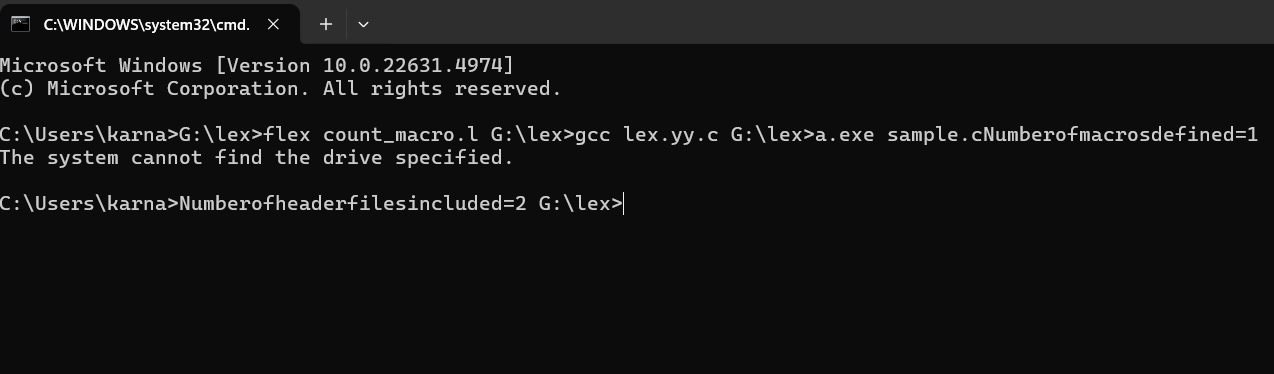
}

intmain(intargc,char\*argv[]){ yyin = fopen(argv[1], "r"); yylex();

printf("Number of macros defined = %d\n", nmacro); printf("Numberofheaderfilesincluded=%d\n",nheader); fclose(yyin);

}

## Output:



# Exp.No.24

## WriteaLEXprogramtoprintallHTMLtagsintheinputfile.

**InputSourceProgram:(sample.html)**

<html>

<body>

<h1>MyFirst Heading</h1>

<p>Myfirstparagraph.</p>

</body>

</html>

## Program:(html.l)

%{

inttags;

%}

%%

"<"[^>]\*>{tags++;printf("%s\n",yytext);}

.|\n{}

%%

intyywrap(void){ return 1; }

intmain(void)

{

FILE\*f;

charfile[10];

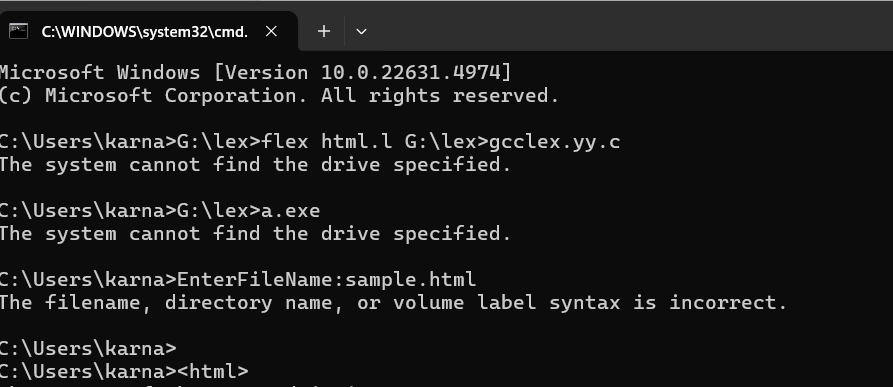
printf("EnterFileName:"); scanf("%s",file);

f=fopen(file,"r"); yyin = f;

yylex();

printf("\nNumberofhtmltags:%d",tags); fclose(yyin);

}

**Output:** 

# Exp.No.25

## Write a LEX program which adds line numbers to the given C program file and display the same in the standard output.

**InputSourceProgram:(sample.c)**

#define PI 3.14 #include<stdio.h> #include<conio.h>void main()

{

inta,b,c=30;

printf("hello");

}

## Program:(addlinenos.l)

%{

intyylineno;

%}

%%

^(.\*)\nprintf("%4d\t%s",++yylineno,yytext);

%%

intyywrap(void){ return 1;

}

intmain(intargc,char\*argv[]){ yyin = fopen(argv[1], "r"); yylex();

fclose(yyin);

}

## Output:

