Ex-1. LEXICAL ANALYZER

S. AKANSHA
RA1911029010060
CSE-CN P1

AIM:
To write a program to implement a lexical analyzer in C.

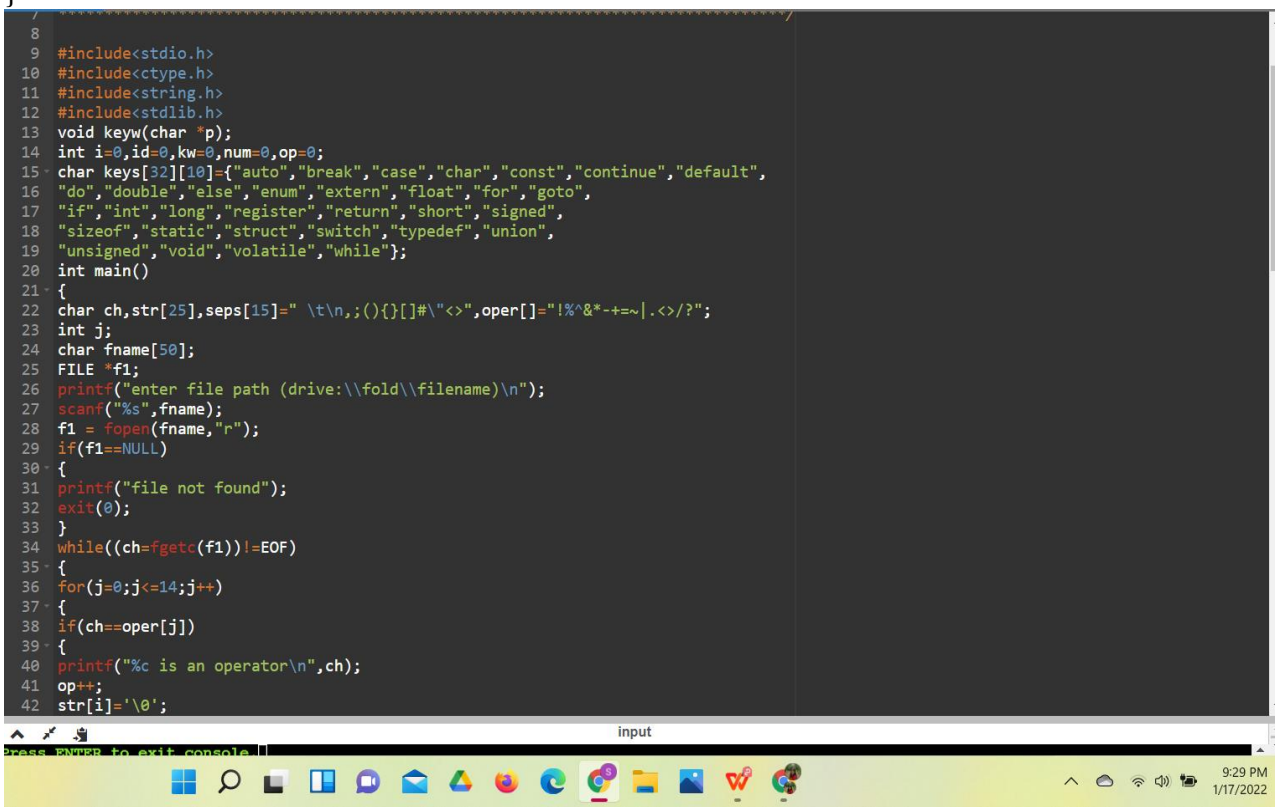
ALGORITHM:
- Start.
- Get the input program from the file sentence.txt.
- Read the program line by line and check if each word in a line is a keyword, identifier,
- math operator, logical operator, numerical value or others symbol.
- For each lexeme read, generate a token as follows:
- If the lexeme is a keyword, then the token is the keyword itself.
- If the lexeme is an identifier, then the token generated is printed on the console as identifier.
- In the same way the math operator, logical operator, numerical values and others symbol are printed on the console.
- The stream of tokens generated are displayed in the console output.
- Stop.

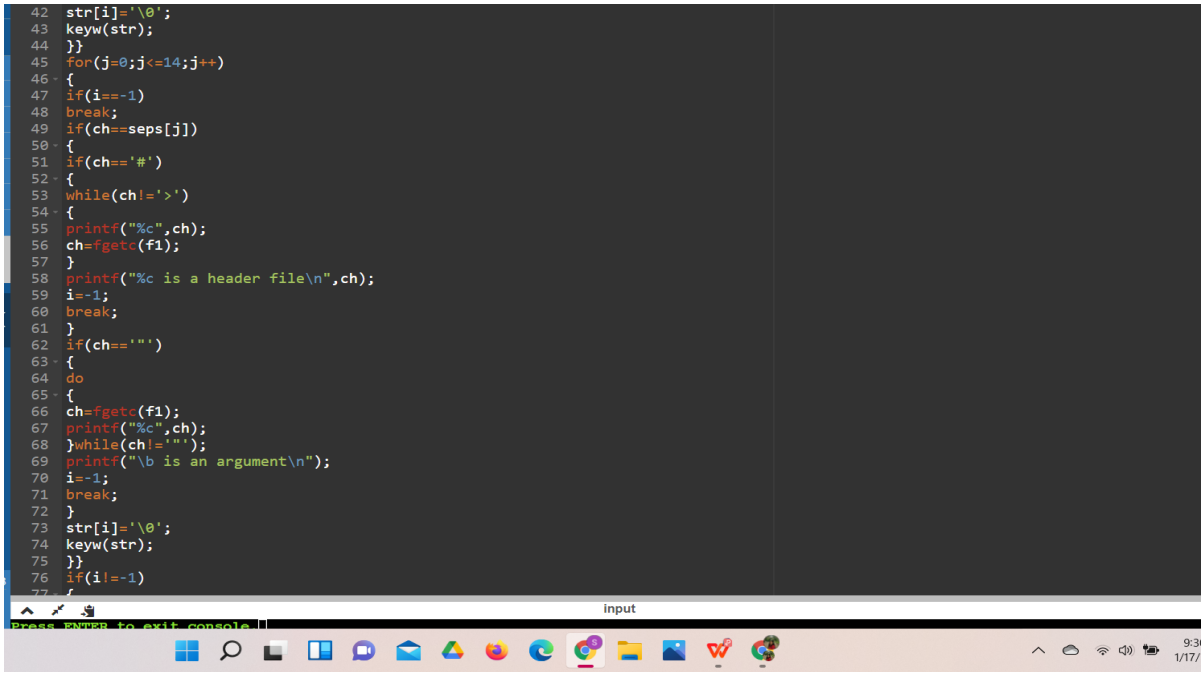
PROGRAM:

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>
#include<stdlib.h>
void keyw(char *p);
int i=0,id=0,kw=0,num=0,op=0;
char keys[32][10]={"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","union",
"unsigned","void","volatile","while"};
int main()
{
char ch,str[25],seps[15]=" \t\n,;(){}[]#\"<>",oper[]="!%^&*-+=~|.<>/?";
int j;
char fname[50];
FILE *f1;
printf("enter file path (drive:\\fold\\filename)\n");
scanf("%s",fname);
f1 = fopen(fname,"r");
if(f1==NULL)
{
printf("file not found");
exit(0);
}
while((ch=fgetc(f1))!=EOF)
{
for(j=0;j<=14;j++)
```

```
{
if(ch==oper[j])
{
printf("%c is an operator\n",ch);
op++;
str[i]='\0';
keyw(str);
}}
for(j=0;j<=14;j++)
{
if(i==-1)
break;
if(ch==seps[j])
{
if(ch=='#')
{
while(ch!='>')
{
printf("%c",ch);
ch=fgetc(f1);
}
printf("%c is a header file\n",ch);
i=-1;
break;
}
if(ch=="")
{
do
{
ch=fgetc(f1);
printf("%c",ch);
}while(ch!="");
printf("\b is an argument\n");
i=-1;
break;
}
str[i]='\0';
keyw(str);
}}
if(i!=-1)
{
str[i]=ch;
i++;
}
else
i=0;
}
printf("Keywords: %d \n",kw);
printf("Identifiers: %d\n",id);
printf("Operators: %d\n",op);
printf("Numbers: %d\n",num);
}
void keyw(char *p)
{
int k,flag=0;
```

```
for(k=0;k<=31;k++)
{
if(strcmp(keys[k],p)==0)
{
printf("%s is a keyword\n",p);
kw++;
flag=1;
break;
}}
if(flag==0)
{
if(isdigit(p[0]))
{
printf("%s is a number\n",p);
num++;
}
else
{
if(p[0]!='\0')
{
printf("%s is an identifier\n",p);
id++;
}}}
i=-1;
}
```

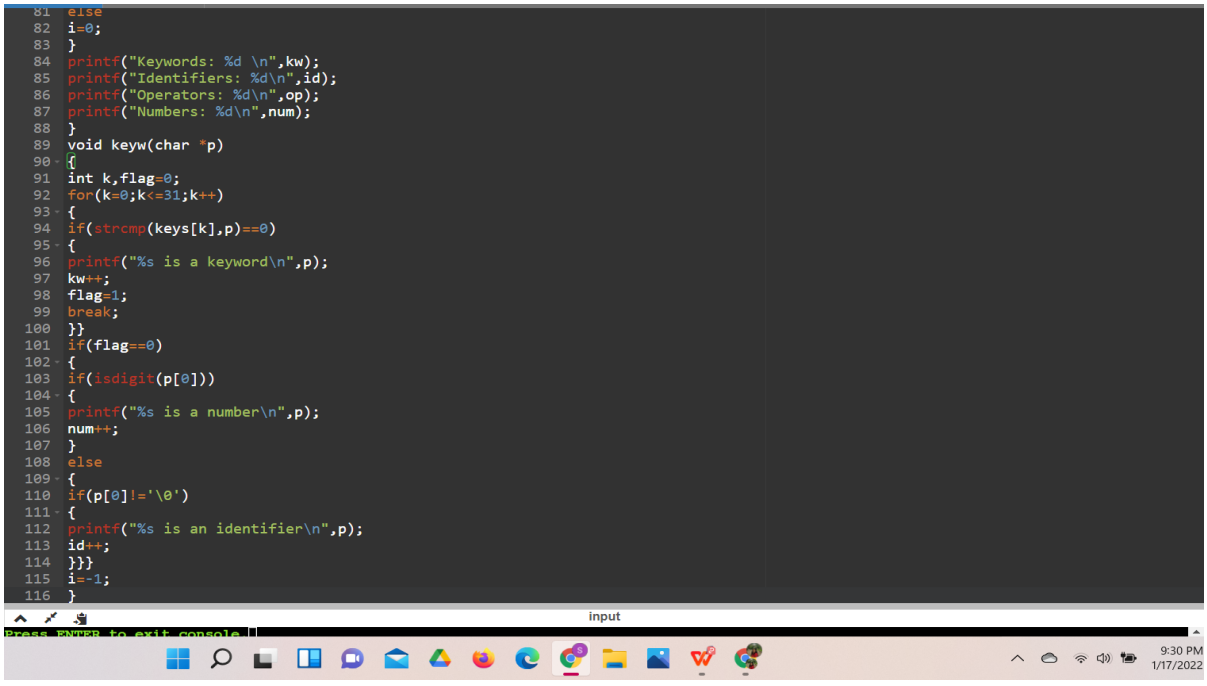




```
42  str[i]='\0';
43  keyw(str);
44  }}
45  for(j=0;j<=14;j++)
46  {
47  if(i==-1)
48  break;
49  if(ch==seps[j])
50  {
51  if(ch=='#')
52  {
53  while(ch!='>')
54  {
55  printf("%c",ch);
56  ch=fgetc(f1);
57  }
58  printf("%c is a header file\n",ch);
59  i=-1;
60  break;
61  }
62  if(ch=='"')
63  {
64  do
65  {
66  ch=fgetc(f1);
67  printf("%c",ch);
68  }while(ch!='"');
69  printf("\b is an argument\n");
70  i=-1;
71  break;
72  }
73  str[i]='\0';
74  keyw(str);
75  }}
76  if(i!=-1)
```

input

Press ENTER to exit console.



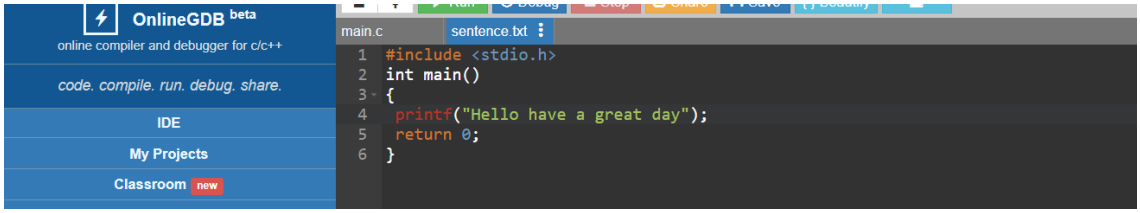
```
81  else
82  i=0;
83  }
84  printf("Keywords: %d \n",kw);
85  printf("Identifiers: %d\n",id);
86  printf("Operators: %d\n",op);
87  printf("Numbers: %d\n",num);
88  }
89  void keyw(char *p)
90  {
91  int k,flag=0;
92  for(k=0;k<=31;k++)
93  {
94  if(strcmp(keys[k],p)==0)
95  {
96  printf("%s is a keyword\n",p);
97  kw++;
98  flag=1;
99  break;
100 }}
101 if(flag==0)
102 {
103 if(isdigit(p[0]))
104 {
105 printf("%s is a number\n",p);
106 num++;
107 }
108 else
109 {
110 if(p[0]!='\0')
111 {
112 printf("%s is an identifier\n",p);
113 id++;
114 }}}
115 i=-1;
116 }
```

input

Press ENTER to exit console.

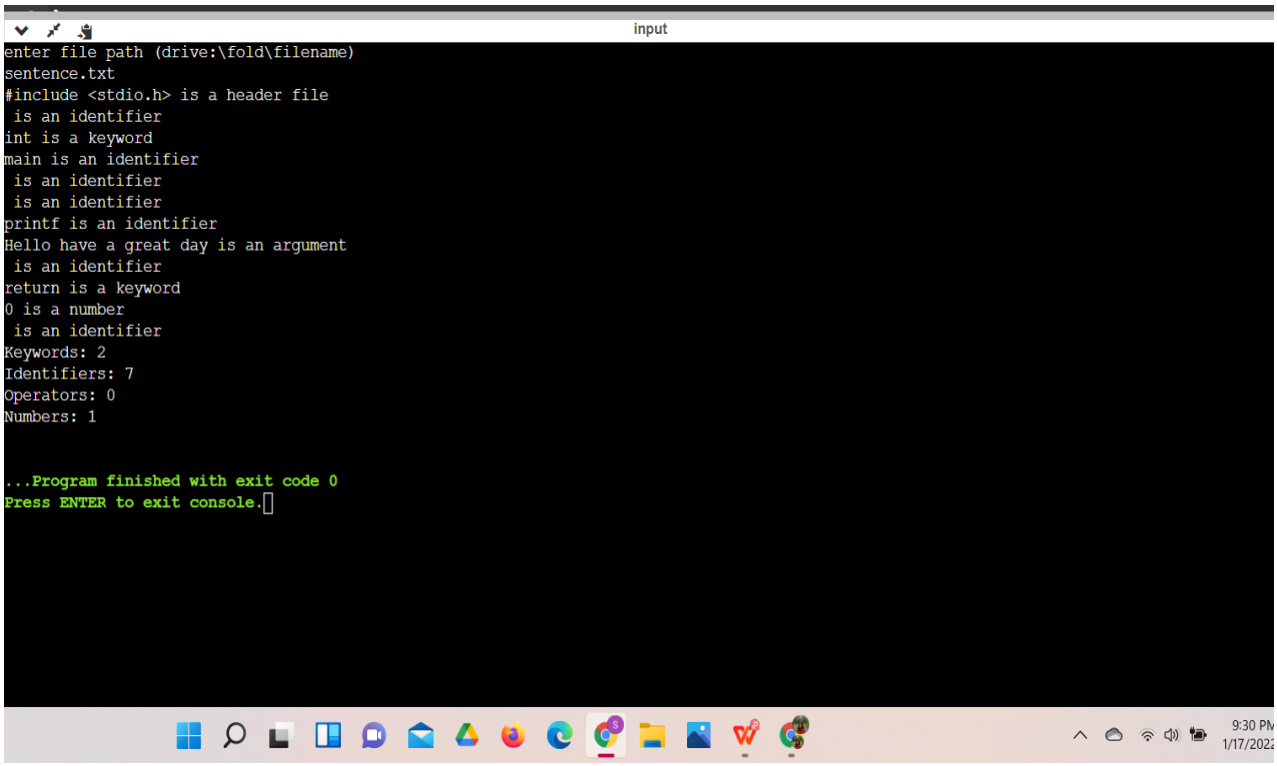
Sentence.txt

```
#include <stdio.h>
int main()
{
 printf("Hello have a great day");
 return 0;
}
```



OUTPUT:

enter file path (drive:\fold\filename)
sentence.txt
#include <stdio.h> is a header file
 is an identifier
int is a keyword
main is an identifier
 is an identifier
 is an identifier
printf is an identifier
Hello have a great day is an argument
 is an identifier
return is a keyword
0 is a number
 is an identifier
Keywords: 2
Identifiers: 7
Operators: 0
Numbers: 1



RESULT:

The implementation of lexical analyzer in C was compiled, executed and verified successfully.

S. AKANSHA
RA1911029010060
P1
EXPERIMENT -2 **CONVERSION FROM REGULAR EXPRESSION TO NFA**

Aim:-

Conversion from Regular Expression to NFA

ALGORITHM:

1. Start
2. Get the input from the user
3. Initialize separate variables and functions for Postfix , Display and NFA
4. Create separate methods for different operators like +,*, .
5. By using Switch case Initialize different cases for the input
6. For ' . ' operator Initialize a separate method by using various stack functions do the same for the other operators like ' * ' and ' + '.
7. Regular expression is in the form like a.b (or) a+b
8. Display the output
9. Stop

Procedure:-

The code for regular expression to NFA has been written in C language and run on a computer.

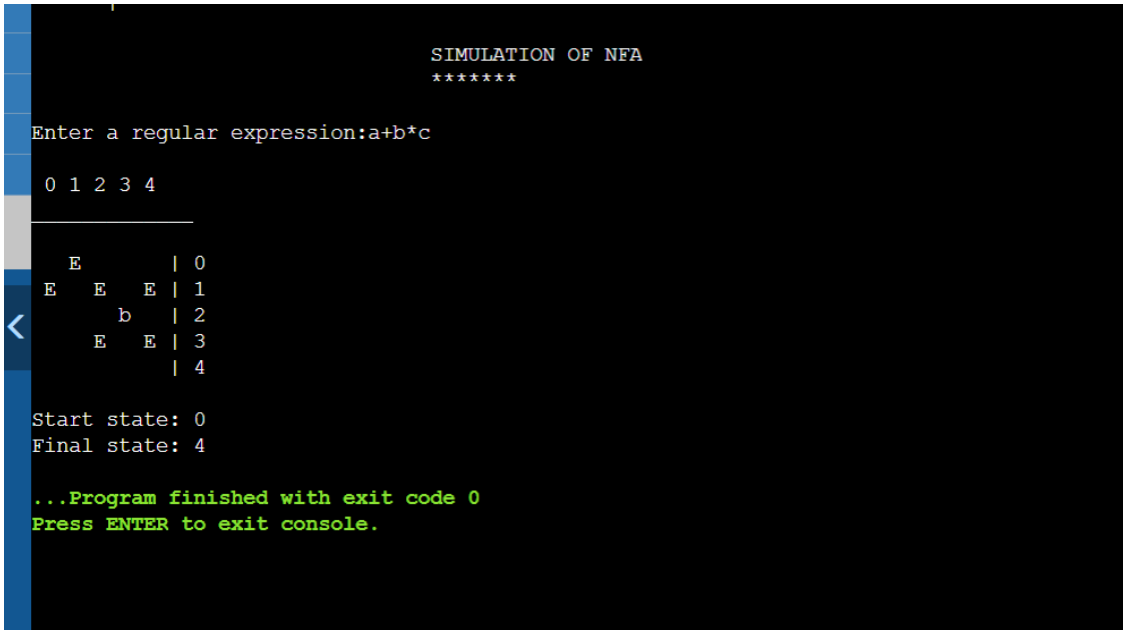
Code:-

```
#include<stdio.h>
#include <string.h>
int main()
{
char m[20],t[10][10];
int n, i , j , r=0, c=0;
printf("\n\t\t\t\tSIMULATION OF NFA");
printf("\n\t\t\t\t********");
for(i=0;i<10;i++)
{
for(j=0;j<10;j++)
{
t[i][j]=' ';
}
}
printf("\n\nEnter a regular expression:");
scanf("%s",m);
n=strlen(m);
for(i=0;i<n;i++)
{
```

```
switch(m[i])
{
case '|' : {
t[r][r+1]='E';
t[r+1][r+2]=m[i-1];
t[r+2][r+5]='E';
t[r][r+3]='E';
t[r+4][r+5]='E';
t[r+3][r+4]=m[i+1];
r=r+5;
break;
}
case '*':{t[r-1][r]='E';
t[r][r+1]='E';
t[r][r+3]='E';
t[r+1][r+2]=m[i-1];
t[r+2][r+1]='E';
t[r+2][r+3]='E';
r=r+3;
break;
}
case '+': {
t[r][r+1]=m[i-1];
t[r+1][r]='E';
r=r+1;
break;
}
default:
{
if(c==0)
{
if((isalpha(m[i]))&&(isalpha(m[i+1])))
{
t[r][r+1]=m[i];
t[r+1][r+2]=m[i+1];
r=r+2;
c=1;
}
c=1;
}
else if(c==1)
{
if(isalpha(m[i+1]))
```

```
{
t[r][r+1]=m[i+1];
r=r+1;
 c=2;
}
}
else
{if(isalpha(m[i+1]))
{
t[r][r+1]=m[i+1];
r=r+1;
c=3;
}
}
}
break;
}
}
printf("\n");
for(j=0;j<=r;j++)
printf(" %d",j);
printf("\n_____\n");
printf("\n");
for(i=0;i<=r;i++)
{
for(j=0;j<=r;j++)
{
printf(" %c",t[i][j]);
}
printf(" | %d",i);
printf("\n");
}
printf("\nStart state: 0\nFinal state: %d",i-1);

}
/*input: a+b*c */
```

Output:-



Result:-

Thus the C program to convert regular expression to NFA has been executed an d the output has been verified successfully.

EX. NO. 3 CONVERSION OF NFA TO DFA

S. Akansha
RA1911029010060

AIM:

To write a program for converting NFA to DFA.

ALGORITHM:

Start
Get the input from the user
Set the only state in SDFA to “unmarked”.
while SDFA contains an unmarked state do:
Let T be that unmarked state
for each a in % do S = e-Closure(MoveNFA(T,a))
if S is not in SDFA already then, add S to SDFA (as an “unmarked” state)
Set MoveDFA(T,a) to S
For each S in SDFA if any s & S is a final state in the NFA then, mark S an a final state in the DFA
Print the result.
Stop the program

PROGRAM:

```
import pandas as pd

# Taking NFA input from User

nfa = {}
n = int(input("No. of states : "))           #Enter total no. of states
t = int(input("No. of transitions : "))      #Enter total no. of transitions/paths eg: a,b so input 2 for a,b,c input 3
for i in range(n):
    state = input("state name : ")           #Enter state name eg: A, B, C, q1, q2 ..etc
    nfa[state] = {}                          #Creating a nested dictionary
    for j in range(t):
        path = input("path : ")              #Enter path eg : a or b in {a,b} 0 or 1 in {0,1}
        print("Enter end state from state {} travelling through path {} : ".format(state,path))
        reaching_state = [x for x in input().split()]  #Enter all the end states that
        nfa[state][path] = reaching_state     #Assigning the end states to the paths in dictionary

print("\nNFA :- \n")
```

```
print(nfa)                              #Printing NFA
print("\nPrinting NFA table :- ")
nfa_table = pd.DataFrame(nfa)
print(nfa_table.transpose())

print("Enter final state of NFA : ")
nfa_final_state = [x for x in input().split()]      # Enter final state/states of NFA
#####################################################

new_states_list = []                        #holds all the new states created in dfa
dfa = {}                                    #dfa dictionary/table or the output structure we
needed
keys_list = list(list(nfa.keys())[0])               #conatins all the states in nfa plus the
states created in dfa are also appended further
path_list = list(nfa[keys_list[0]].keys())    #list of all the paths eg: [a,b] or [0,1]

#####################################################

# Computing first row of DFA transition table

dfa[keys_list[0]] = {}                      #creating a nested dictionary in dfa
for y in range(t):
   var = "".join(nfa[keys_list[0]][path_list[y]])   #creating a single string from all the
elements of the list which is a new state
   dfa[keys_list[0]][path_list[y]] = var          #assigning the state in DFA table
   if var not in keys_list:                    #if the state is newly created
      new_states_list.append(var)              #then append it to the new_states_list
      keys_list.append(var)                    #as well as to the keys_list which contains
all the states

#####################################################

# Computing the other rows of DFA transition table

while len(new_states_list) != 0:                #consition is true only if the
new_states_list is not empty
   dfa[new_states_list[0]] = {}                 #taking the first element of the
new_states_list and examining it
   for _ in range(len(new_states_list[0])):
      for i in range(len(path_list)):
         temp = []                              #creating a temporay list
         for j in range(len(new_states_list[0])):
            temp += nfa[new_states_list[0][j]][path_list[i]]  #taking the union of the
states
```

```
        s = ""
        s = s.join(temp)                    #creating a single string(new state) from all
the elements of the list
        if s not in keys_list:              #if the state is newly created
            new_states_list.append(s)           #then append it to the new_states_list
            keys_list.append(s)             #as well as to the keys_list which contains
all the states
        dfa[new_states_list[0]][path_list[i]] = s   #assigning the new state in the DFA
table

    new_states_list.remove(new_states_list[0])      #Removing the first element in the
new_states_list

print("\nDFA :- \n")
print(dfa)                                  #Printing the DFA created
print("\nPrinting DFA table :- ")
dfa_table = pd.DataFrame(dfa)
print(dfa_table.transpose())

dfa_states_list = list(dfa.keys())
dfa_final_states = []
for x in dfa_states_list:
    for i in x:
        if i in nfa_final_state:
            dfa_final_states.append(x)
            break

print("\nFinal states of the DFA are : ",dfa_final_states)      #Printing Final states of
DFA
```

OUTPUT:

```
No. of states : 4
No. of transitions : 2
state name : A
path : a
Enter end state from state A travelling through path a :
A B
path : b
Enter end state from state A travelling through path b :
A
```

state name : B
path : a
Enter end state from state B travelling through path a :
C
path : b
Enter end state from state B travelling through path b :
C
state name : C
path : a
Enter end state from state C travelling through path a :
D
path : b
Enter end state from state C travelling through path b :
D
state name : D
path : a
Enter end state from state D travelling through path a :

path : b
Enter end state from state D travelling through path b :

NFA :-

{'A': {'a': ['A', 'B'], 'b': ['A']}, 'B': {'a': ['C'], 'b': ['C']}, 'C': {'a': ['D'], 'b': ['D']}, 'D': {'a': [], 'b': []}}

Printing NFA table :-

	a	b
A	[A, B]	[A]
B	[C]	[C]
C	[D]	[D]
D	[]	[]

Enter final state of NFA :
D

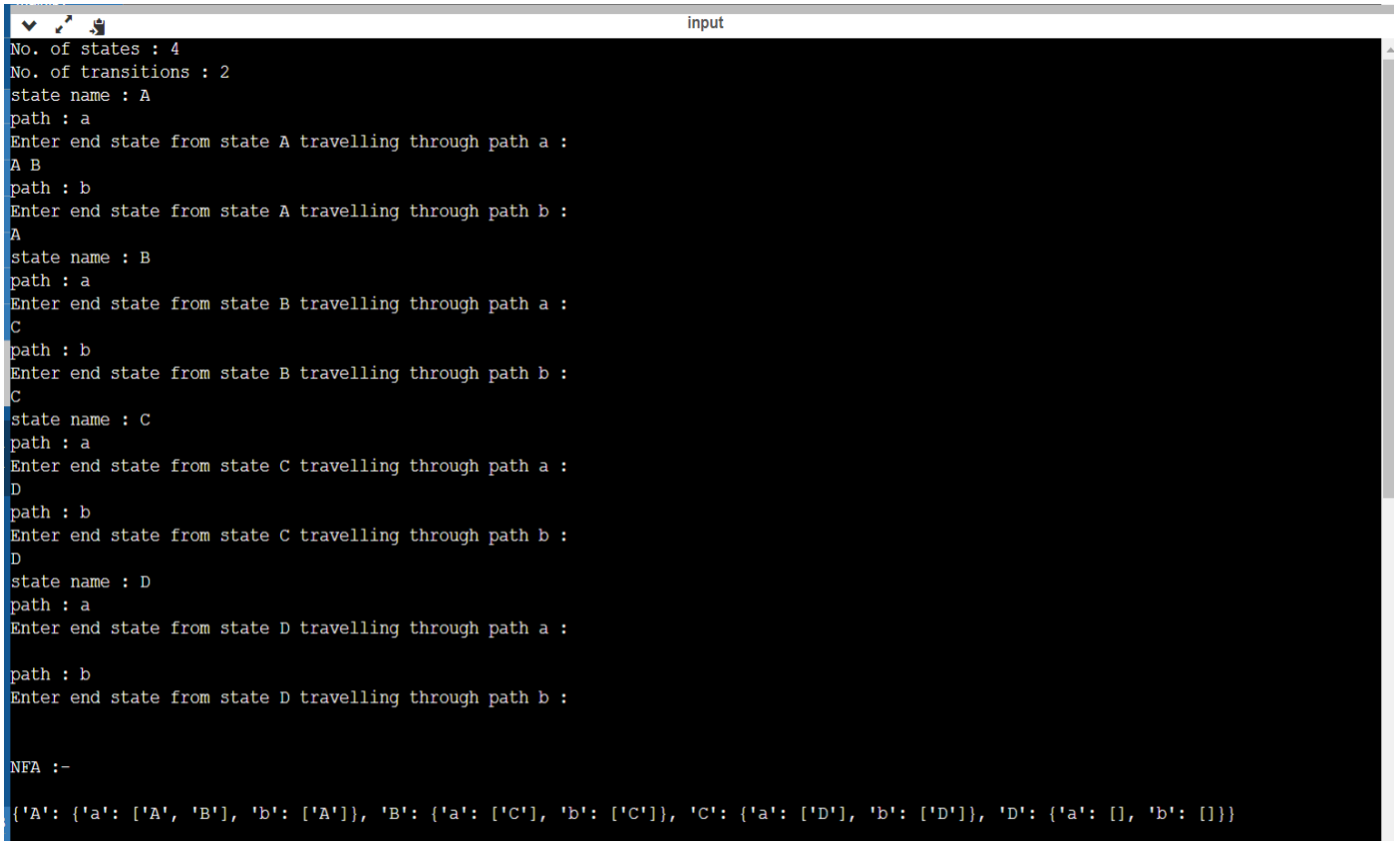
DFA :-

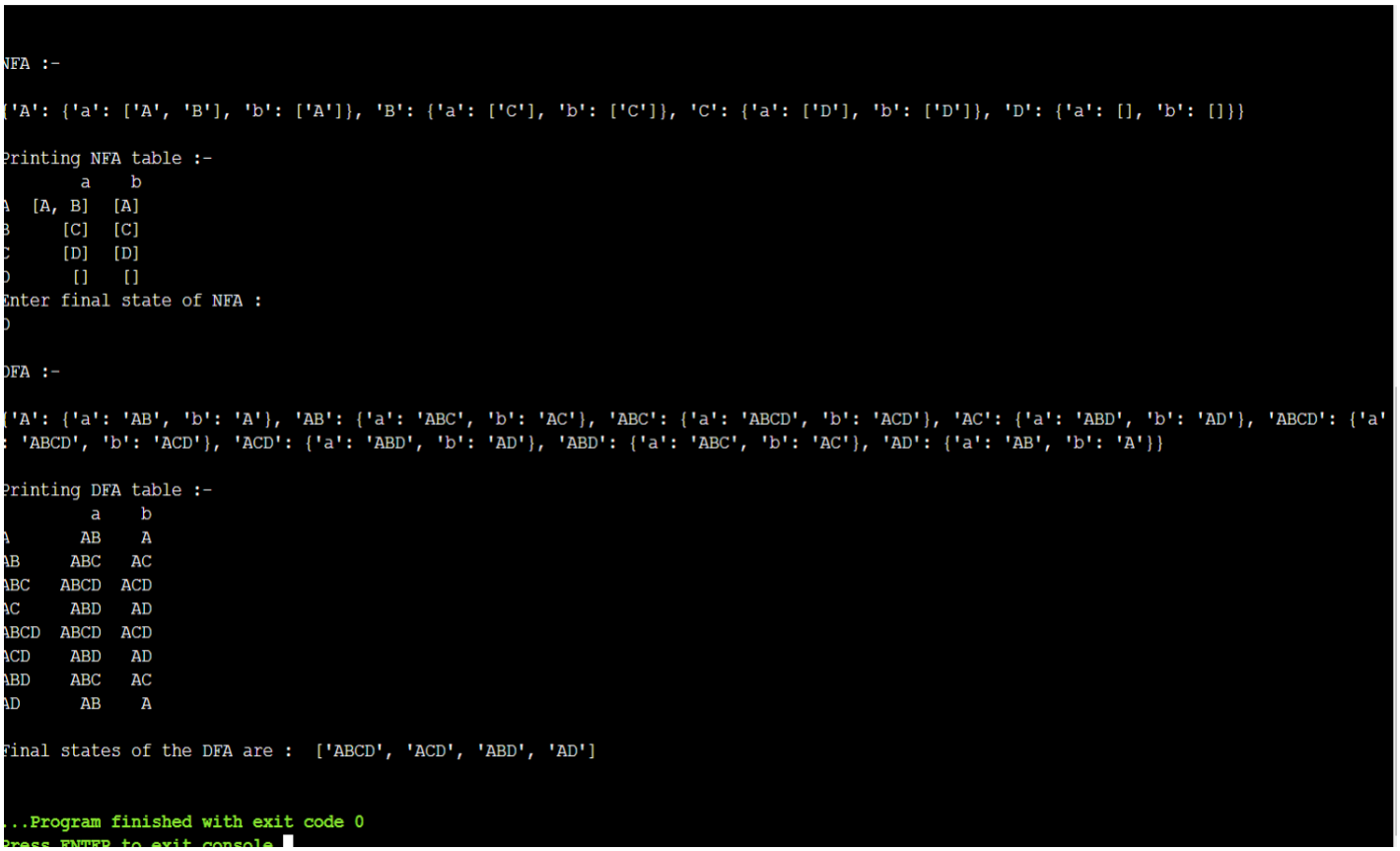
{'A': {'a': 'AB', 'b': 'A'}, 'AB': {'a': 'ABC', 'b': 'AC'}, 'ABC': {'a': 'ABCD', 'b': 'ACD'}, 'AC': {'a': 'ABD', 'b': 'AD'}, 'ABCD': {'a': 'ABCD', 'b': 'ACD'}, 'ACD': {'a': 'ABD', 'b': 'AD'}, 'ABD': {'a': 'ABC', 'b': 'AC'}, 'AD': {'a': 'AB', 'b': 'A'}}

Printing DFA table :-

	a	b
A	AB	A
AB	ABC	AC
ABC	ABCD	ACD
AC	ABD	AD
ABCD	ABCD	ACD
ACD	ABD	AD
ABD	ABC	AC
AD	AB	A

Final states of the DFA are : ['ABCD', 'ACD', 'ABD', 'AD']





RESULT :

The given NFA was converted to a DFA using python successfully.

S. AKANSHA
RA1911029010060 CSE-CN P1

EX. NO. 4(a)- ELIMINATION OF LEFT RECURSION

AIM:

A program for Elimination of Left Recursion.

ALGORITHM:

- Start the program.
- Initialize the arrays for taking input from the user.
- Prompt the user to input the no. of non-terminals having left recursion and no. of productions for these non-terminals.
- Prompt the user to input the production for non-terminals.
- Eliminate left recursion using the following rules:- A->Aα1| Aα2 | …. |Aαm
  A->β1| β2| …. | βn
- Then replace it by
  A-> βi A' i=1,2,3,… m
  A'-> αj A' j=1,2,3,… n
  A'-> Ɛ
- After eliminating the left recursion by applying these rules, display the productions without left recursion.
- Stop.

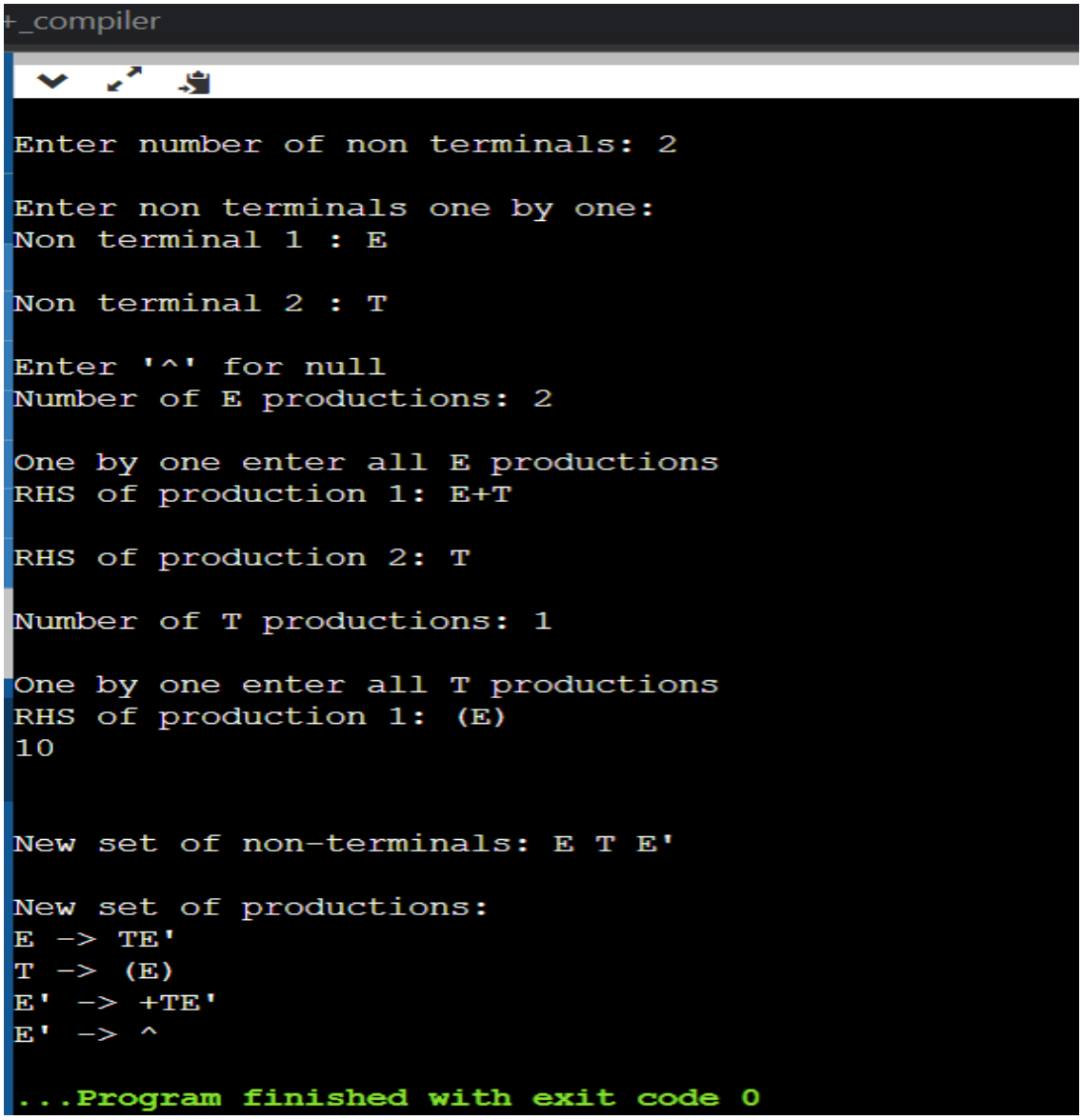
PROGRAM:

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;

int main()
{
int n;
cout<<"\nEnter number of non terminals: ";
cin>>n;
cout<<"\nEnter non terminals one by one: ";
int i;
vector<string> nonter(n);
vector<int> leftrecr(n,0);
for(i=0;i<n;++i) {
cout<<"\nNon terminal "<<i+1<<" : ";
cin>>nonter[i];
}
```

```
vector<vector<string> > prod;
cout<<"\nEnter '^' for null";
for(i=0;i<n;++i)
{
cout<<"\nNumber of "<<nonter[i]<<" productions: ";
int k;
cin>>k;
int j;
cout<<"\nOne by one enter all "<<nonter[i]<<" productions";
vector<string> temp(k);
for(j=0;j<k;++j)
{
cout<<"\nRHS of production "<<j+1<<": ";
string abc;
cin>>abc;
temp[j]=abc;
if(nonter[i].length()<=abc.length()&&nonter[i].compare(abc.substr(0,nonter[i].lengt
   h()))==0)
        leftrecr[i]=1;
}
prod.push_back(temp);
}
for(i=0;i<n;++i)
{
cout<<leftrecr[i];
}
for(i=0;i<n;++i)
{
if(leftrecr[i]==0)
continue;
int j;
nonter.push_back(nonter[i]+"'");
vector<string> temp;
for(j=0;j<prod[i].size();++j)
{
if(nonter[i].length()<=prod[i][j].length()&&nonter[i].compare(prod[i][j].substr(0,no
nter[i].length ()))==0)
{
string abc=prod[i][j].substr(nonter[i].length(),prod[i][j].length()-
nonter[i].length())+nonter[i]+"'";
temp.push_back(abc);
prod[i].erase(prod[i].begin()+j);
--j;
}
```

```
else
{
prod[i][j]+=nonter[i]+"'";
}
}
temp.push_back("^"); prod.push_back(temp);
}
cout<<"\n\n";
cout<<"\nNew set of non-terminals: ";
for(i=0;i<nonter.size();++i)
cout<<nonter[i]<<" ";
cout<<"\n\nNew set of productions: ";
for(i=0;i<nonter.size();++i)
{
int j; for(j=0;j<prod[i].size();++j)
{
cout<<"\n"<<nonter[i]<<" -> "<<prod[i][j];
}
}
return 0;
}
```

OUTPUT :



RESULT :
A program for Elimination of Left Recursion was run successfully.

S.AKANSHA
RA1911029010060
CSE-CN P1

EX. NO. 4(b)-LEFT FACTORING

AIM :

A program for implementation Of Left Factoring

ALGORITHM :

- Start
- Ask the user to enter the set of productions.
- Check for common symbols in the given set of productions by comparing with: A->aB1|aB2
- If found, replace the particular productions with: A->aA' A'->B1 | B2|ε
- Display the output.
- Exit

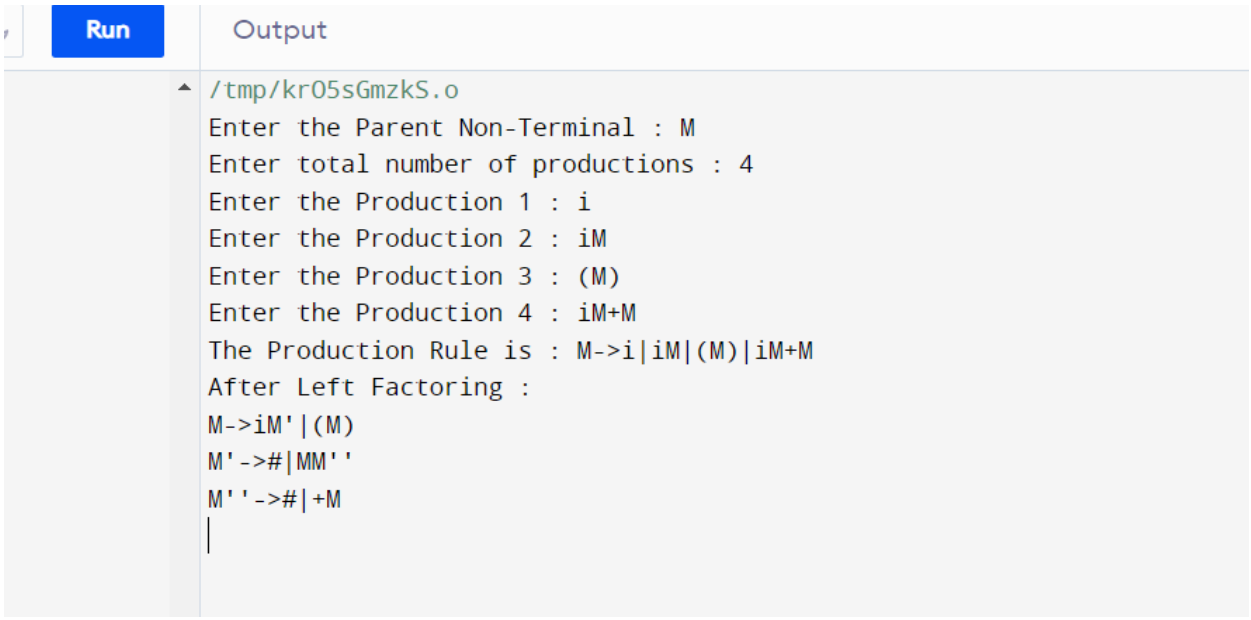
CODE :

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
        int n,j,l,i,m;
        int len[10] = {};
   string a, b1, b2, flag;
   char c;
   cout << "Enter the Parent Non-Terminal : ";
   cin >> c;
   a.push_back(c);
   b1 += a + "\'->";
   b2 += a + "\'\'->";;
   a += "->";
   cout << "Enter total number of productions : ";
   cin >> n;
   for (i = 0; i < n; i++)
   {
      cout << "Enter the Production " << i + 1 << " : ";
      cin >> flag;
      len[i] = flag.size();
      a += flag;
```

```
    if (i != n - 1)
    {
    a += "|";
                }
}
cout << "The Production Rule is : " << a << endl;
char x = a[3];
for (i = 0, m = 3; i < n; i++)
{
    if (x != a[m])
    {
         while (a[m++] != '|');
    }
    else
    {
       if (a[m + 1] != '|')
        {
           b1 += "|" + a.substr(m + 1, len[i] - 1);
           a.erase(m - 1, len[i] + 1);
        }
        else
        {
              b1 += "#";
           a.insert(m + 1, 1, a[0]);
           a.insert(m + 2, 1, '\');
           m += 4;
        }
    }
}
char y = b1[6];
for (i = 0, m = 6; i < n - 1; i++)
{
    if (y == b1[m])
    {
        if (b1[m + 1] != '|')
        {
           flag.clear();
           for (int s = m + 1; s < b1.length(); s++)
           {
               flag.push_back(b1[s]);
                              }
           b2 += "|" + flag;
           b1.erase(m - 1, flag.length() + 2);
        }
```

```
        else
        {
            b1.insert(m + 1, 1, b1[0]);
          b1.insert(m + 2, 2, '\');
          b2 += "#";
          m += 5;
        }
      }
  }
  b2.erase(b2.size() - 1);
  cout << "After Left Factoring : " << endl;
  cout << a << endl;
  cout << b1 << endl;
  cout << b2 << endl;
  return 0;
  }
```

OUTPUT :



RESULT :

A program for implementation Of Left Factoring was compiled and run successfully

Ex-5: Computation of FIRST and FOLLOW

T. AKANSHA
RA1911029010060
CSE-CN P1

AIM:
To write a program for Computation of FIRST and FOLLOW.

ALGORITHM:

For computing the first:
1. If X is a terminal then FIRST(X) = {X}
Example: F -> I | id
We can write it as FIRST(F) -> { (, id)
2. If X is a non-terminal like E -> T then to get FIRSTI substitute T with other productions until you get a terminal as the first symbol
3. If X -> ε then add ε to FIRST(X).

For computing the follow:
1. Always check the right side of the productions for a non-terminal, whose FOLLOW set is being found. (never see the left side).
2. (a) If that non-terminal (S,A,B…) is followed by any terminal (a,b…,*,+,(,)…) , then add that terminal into the FOLLOW set.
(b) If that non-terminal is followed by any other non-terminal then add FIRST of other nonterminal into the FOLLOW set.

PROGRAM:

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

int i,j,l,m,n=0,o,p,nv,z=0,x=0;
char str[10],temp,temp2[10],temp3[20],*ptr;

struct prod
{
   char lhs[10],rhs[10][10],ft[10],fol[10];
   int n;
}pro[10];

void findter()
{
   int k,t;
   for(k=0;k<n;k++)
   {
      if(temp==pro[k].lhs[0])
      {
         for(t=0;t<pro[k].n;t++)
         {
            if( pro[k].rhs[t][0]<65 || pro[k].rhs[t][0]>90 )
               pro[i].ft[strlen(pro[i].ft)]=pro[k].rhs[t][0];
```

```
            else if( pro[k].rhs[t][0]>=65 && pro[k].rhs[t][0]<=90 )
            {
               temp=pro[k].rhs[t][0];
               if(temp=='S')
                  pro[i].ft[strlen(pro[i].ft)]='#';
               findter();
            }
         }
         break;
      }
   }
}

void findfol()
{
   int k,t,p1,o1,chk;
   char *ptr1;
   for(k=0;k<n;k++)
   {
      chk=0;
      for(t=0;t<pro[k].n;t++)
      {
         ptr1=strchr(pro[k].rhs[t],temp);
         if( ptr1 )
         {
            p1=ptr1-pro[k].rhs[t];
            if(pro[k].rhs[t][p1+1]>=65 && pro[k].rhs[t][p1+1]<=90)
            {
               for(o1=0;o1<n;o1++)
                  if(pro[o1].lhs[0]==pro[k].rhs[t][p1+1])
                  {
                        strcat(pro[i].fol,pro[o1].ft);
                        chk++;
                  }
            }
            else if(pro[k].rhs[t][p1+1]=='\0')
            {
               temp=pro[k].lhs[0];
               if(pro[l].rhs[j][p]==temp)
                  continue;
               if(temp=='S')
                  strcat(pro[i].fol,"$");
               findfol();
               chk++;
            }
            else
            {
               pro[i].fol[strlen(pro[i].fol)]=pro[k].rhs[t][p1+1];
               chk++;
            }
         }
      }
      if(chk>0)
         break;
   }
```

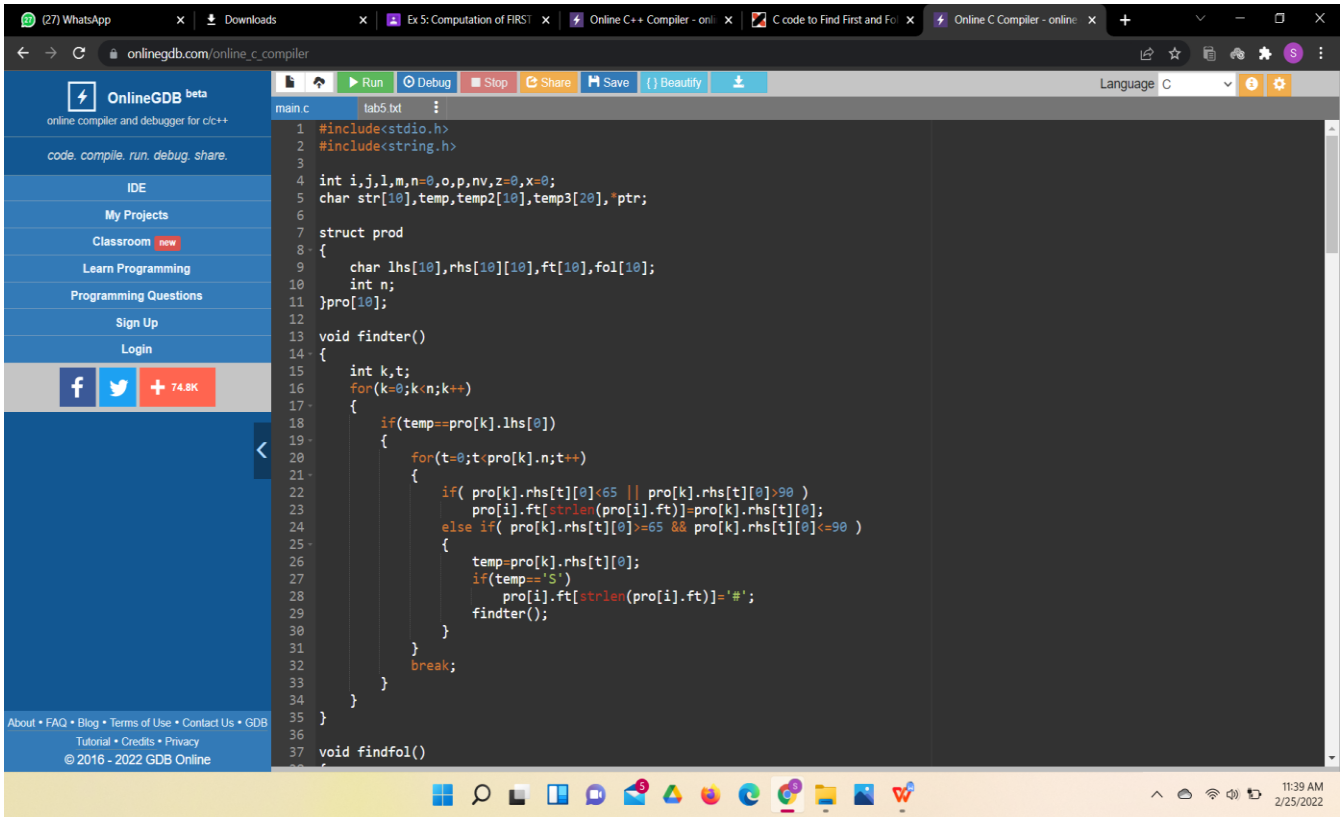
```
}

int main()
{
   FILE *f;
   //clrscr();

   for(i=0;i<10;i++)
      pro[i].n=0;

   f=fopen("tab5.txt","r");
   while(!feof(f))
   {
      fscanf(f,"%s",pro[n].lhs);
      if(n>0)
      {
         if( strcmp(pro[n].lhs,pro[n-1].lhs) == 0 )
         {
            pro[n].lhs[0]='\0';
            fscanf(f,"%s",pro[n-1].rhs[pro[n-1].n]);
            pro[n-1].n++;
            continue;
         }
      }
      fscanf(f,"%s",pro[n].rhs[pro[n].n]);
      pro[n].n++;
      n++;
   }

   printf("\n\nTHE GRAMMAR IS AS FOLLOWS\n\n");
   for(i=0;i<n;i++)
      for(j=0;j<pro[i].n;j++)
         printf("%s -> %s\n",pro[i].lhs,pro[i].rhs[j]);

   pro[0].ft[0]='#';
   for(i=0;i<n;i++)
   {
      for(j=0;j<pro[i].n;j++)
      {
         if( pro[i].rhs[j][0]<65 || pro[i].rhs[j][0]>90 )
         {
            pro[i].ft[strlen(pro[i].ft)]=pro[i].rhs[j][0];
         }
         else if( pro[i].rhs[j][0]>=65 && pro[i].rhs[j][0]<=90 )
         {
            temp=pro[i].rhs[j][0];
            if(temp=='S')
               pro[i].ft[strlen(pro[i].ft)]='#';
            findter();
         }
      }
   }

   printf("\n\nFIRST\n");
   for(i=0;i<n;i++)
```

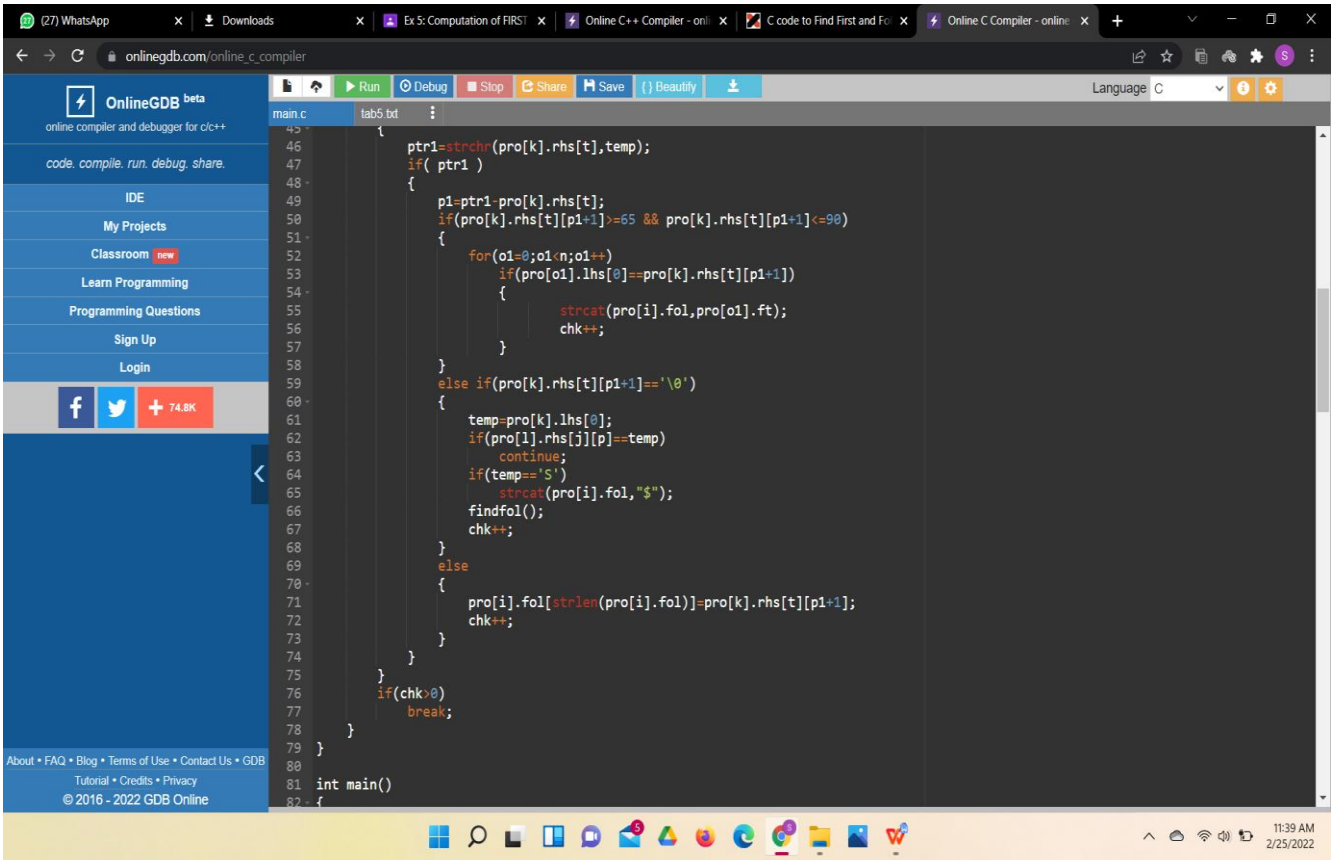
```
{
   printf("\n%s -> ",pro[i].lhs);
   for(j=0;j<strlen(pro[i].ft);j++)
   {
      for(l=j-1;l>=0;l--)
         if(pro[i].ft[l]==pro[i].ft[j])
            break;
      if(l==-1)
         printf("%c",pro[i].ft[j]);
   }
}

for(i=0;i<n;i++)
   temp2[i]=pro[i].lhs[0];
pro[0].fol[0]='$';
for(i=0;i<n;i++)
{
   for(l=0;l<n;l++)
   {
      for(j=0;j<pro[i].n;j++)
      {
         ptr=strchr(pro[l].rhs[j],temp2[i]);
         if( ptr )
         {
            p=ptr-pro[l].rhs[j];
            if(pro[l].rhs[j][p+1]>=65 && pro[l].rhs[j][p+1]<=90)
            {
               for(o=0;o<n;o++)
                  if(pro[o].lhs[0]==pro[l].rhs[j][p+1])
                        strcat(pro[i].fol,pro[o].ft);
            }
            else if(pro[l].rhs[j][p+1]=='\0')
            {
               temp=pro[l].lhs[0];
               if(pro[l].rhs[j][p]==temp)
                  continue;
               if(temp=='S')
                  strcat(pro[i].fol,"$");
               findfol();
            }
            else
               pro[i].fol[strlen(pro[i].fol)]=pro[l].rhs[j][p+1];
         }
      }
   }
}

printf("\n\nFOLLOW\n");
for(i=0;i<n;i++)
{
   printf("\n%s -> ",pro[i].lhs);
   for(j=0;j<strlen(pro[i].fol);j++)
   {
      for(l=j-1;l>=0;l--)
         if(pro[i].fol[l]==pro[i].fol[j])
```

```
                break;
            if(l==-1)
                printf("%c",pro[i].fol[j]);
        }
    }
    printf("\n");
    //getch();
}
```

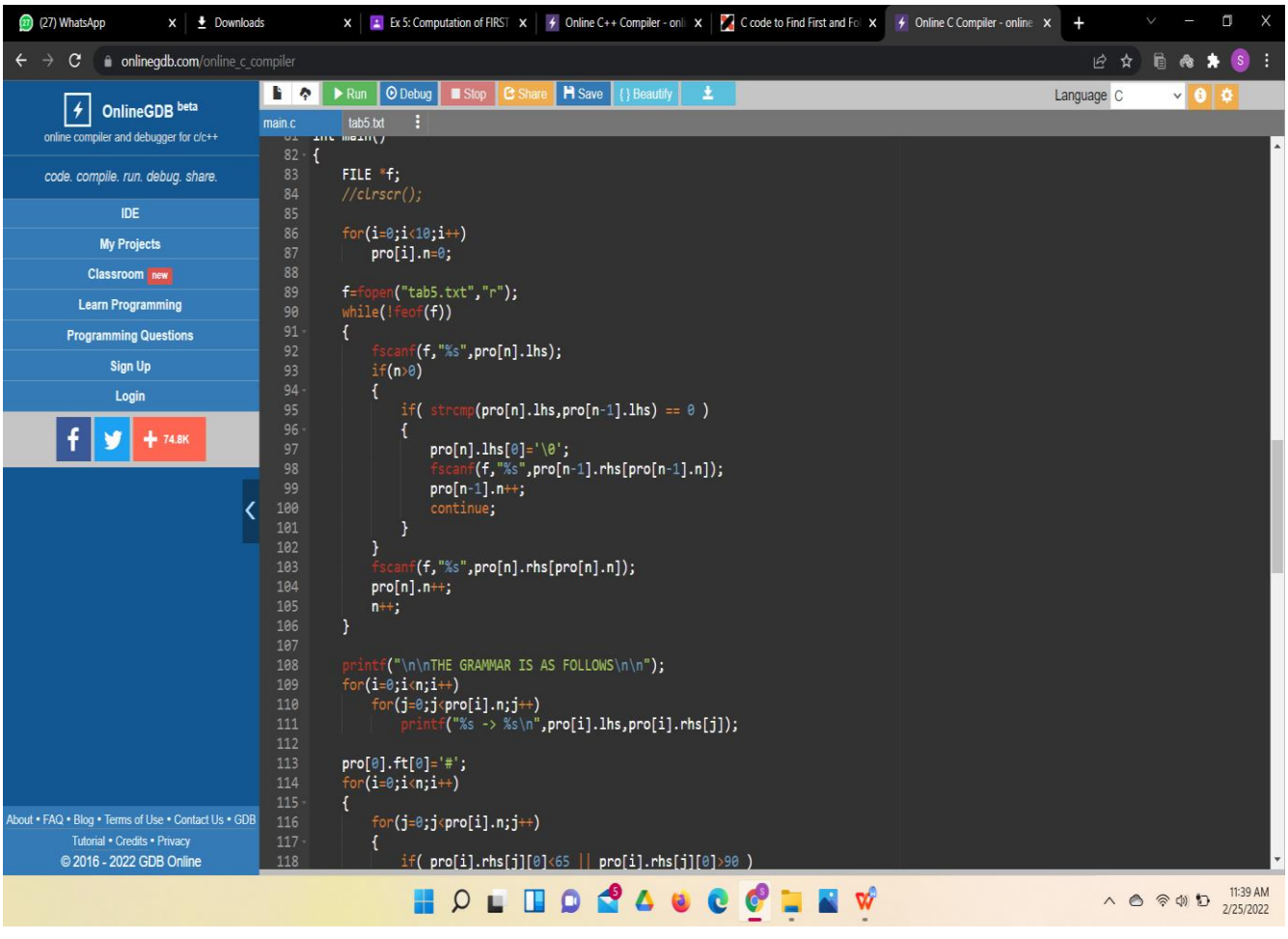
tab5.txt:

```
S ABE
S a
A p
A t
B Aq
S f
A w
```

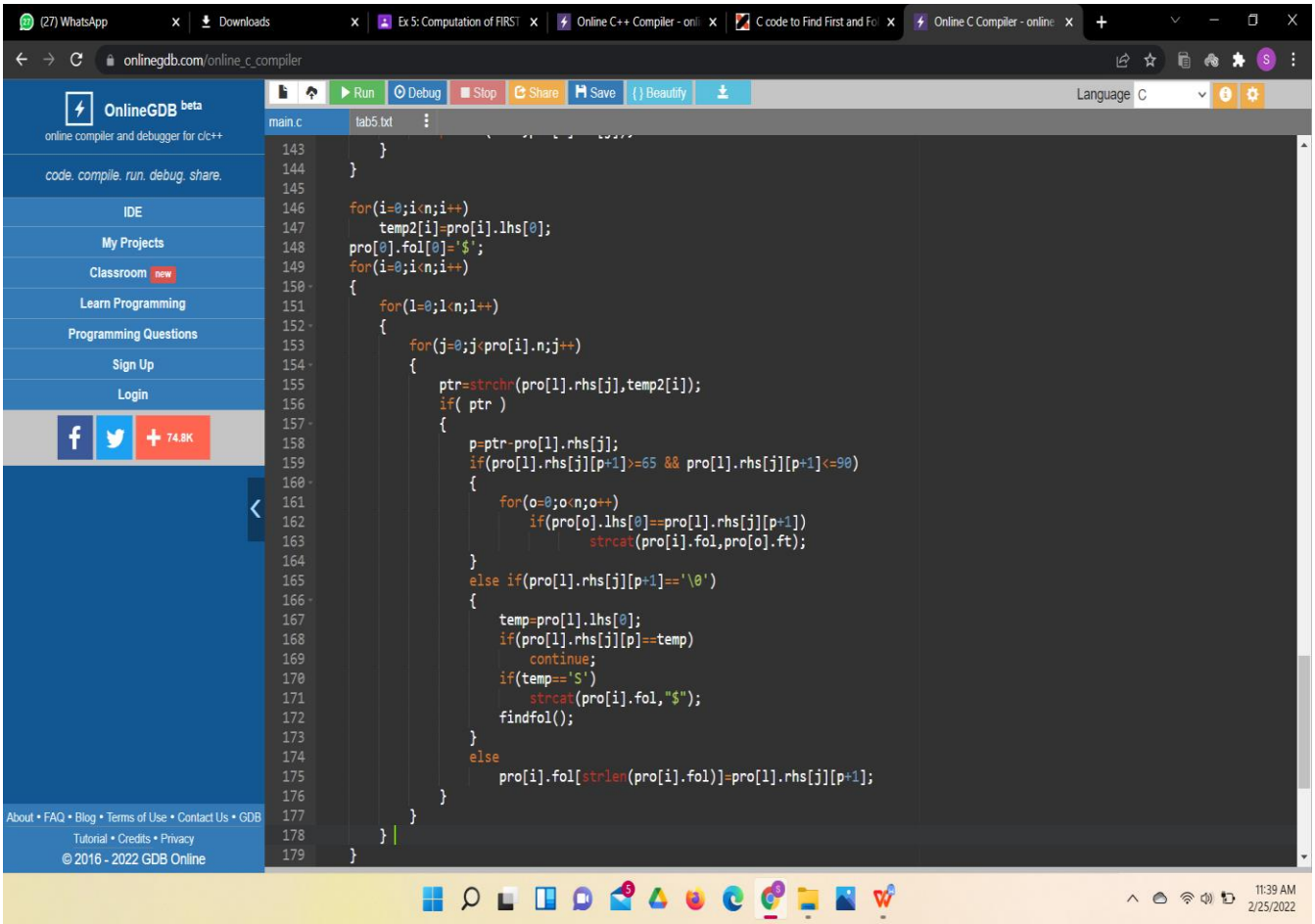


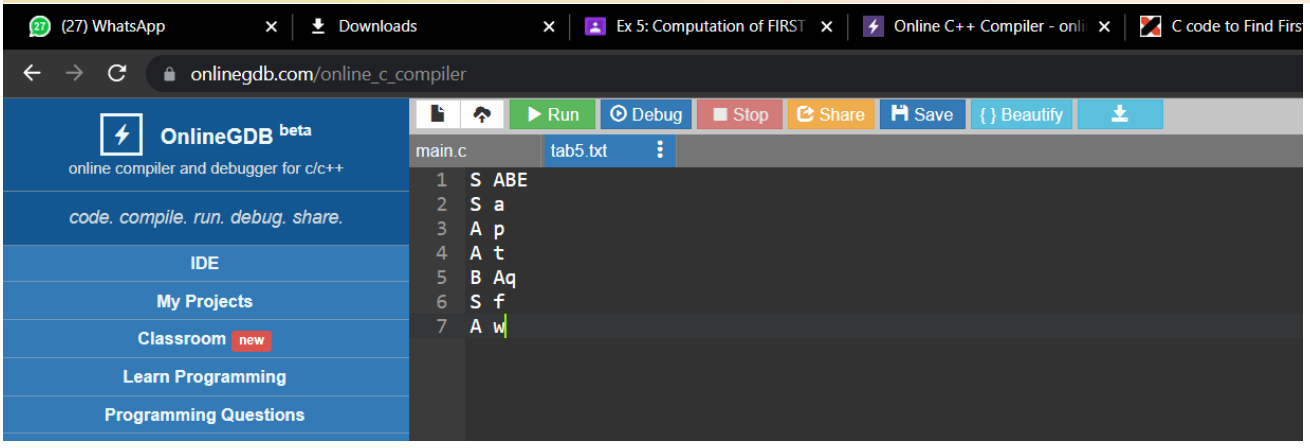


```
        {
            ptr1=strchr(pro[k].rhs[t],temp);
            if( ptr1 )
            {
                p1=ptr1-pro[k].rhs[t];
                if(pro[k].rhs[t][p1+1]>=65 && pro[k].rhs[t][p1+1]<=90)
                {
                    for(o1=0;o1<n;o1++)
                        if(pro[o1].lhs[0]==pro[k].rhs[t][p1+1])
                        {
                                strcat(pro[i].fol,pro[o1].ft);
                                chk++;
                        }
                }
                else if(pro[k].rhs[t][p1+1]=='\0')
                {
                    temp=pro[k].lhs[0];
                    if(pro[l].rhs[j][p]==temp)
                        continue;
                    if(temp=='S')
                        strcat(pro[i].fol,"$");
                    findfol();
                    chk++;
                }
                else
                {
                    pro[i].fol[strlen(pro[i].fol)]=pro[k].rhs[t][p1+1];
                    chk++;
                }
            }
        }
        if(chk>0)
            break;
    }
}

int main()
{
```

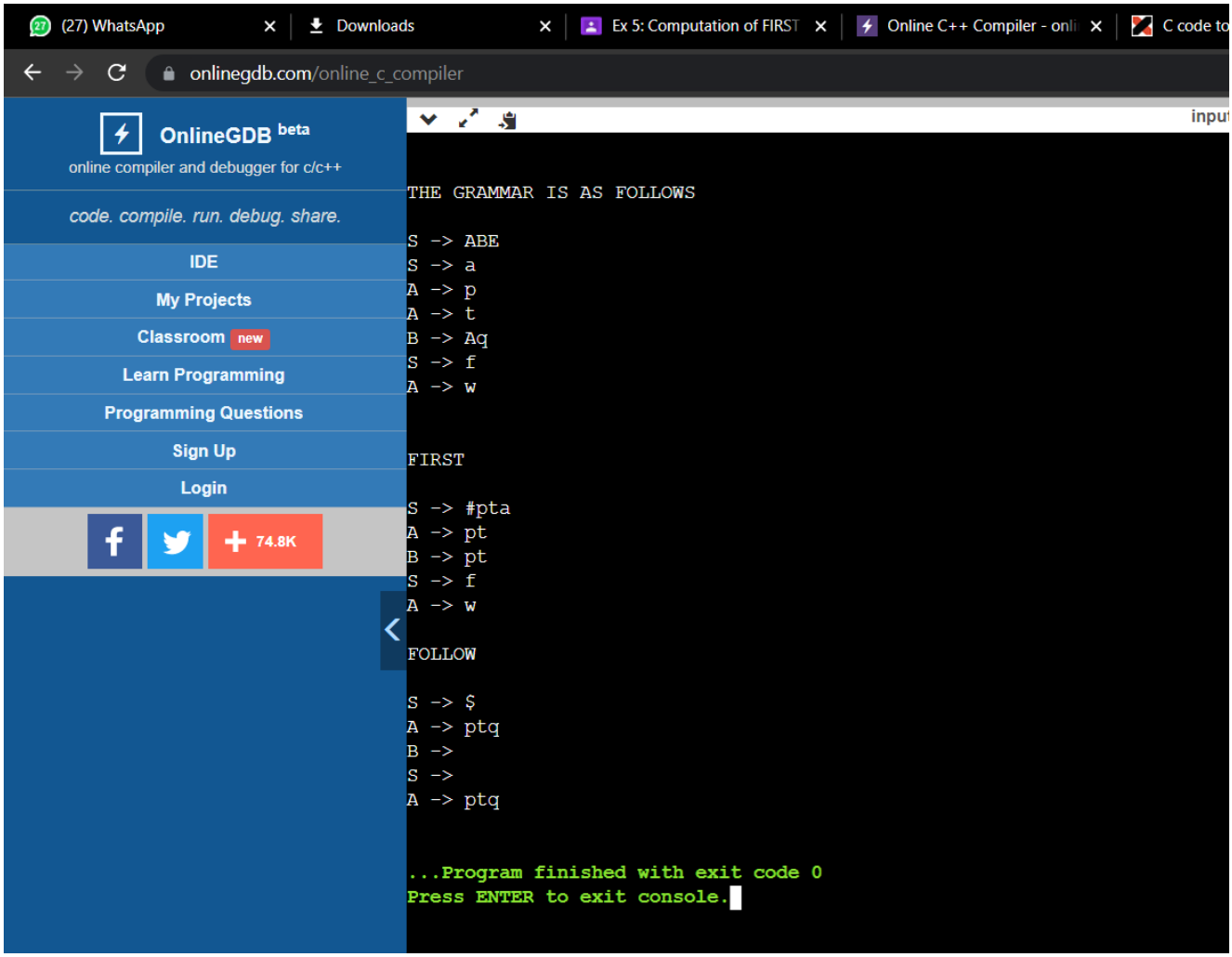


```
{
    FILE *f;
    //clrscr();

    for(i=0;i<10;i++)
        pro[i].n=0;

    f=fopen("tab5.txt","r");
    while(!feof(f))
    {
        fscanf(f,"%s",pro[n].lhs);
        if(n>0)
        {
            if( strcmp(pro[n].lhs,pro[n-1].lhs) == 0 )
            {
                pro[n].lhs[0]='\0';
                fscanf(f,"%s",pro[n-1].rhs[pro[n-1].n]);
                pro[n-1].n++;
                continue;
            }
        }
        fscanf(f,"%s",pro[n].rhs[pro[n].n]);
        pro[n].n++;
        n++;
    }

    printf("\n\nTHE GRAMMAR IS AS FOLLOWS\n\n");
    for(i=0;i<n;i++)
        for(j=0;j<pro[i].n;j++)
            printf("%s -> %s\n",pro[i].lhs,pro[i].rhs[j]);

    pro[0].ft[0]='#';
    for(i=0;i<n;i++)
    {
        for(j=0;j<pro[i].n;j++)
        {
            if( pro[i].rhs[j][0]<65 || pro[i].rhs[j][0]>90 )
```


(27) WhatsApp
Downloads
Ex 5: Computation of FIRST
Online C++ Compiler - onli
C code to Find First and Fo
Online C Compiler - online
onlinegdb.com/online_c_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
74.8K
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2022 GDB Online
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
tab5.txt
}
}
for(i=0;i<n;i++)
temp2[i]=pro[i].lhs[0];
pro[0].fol[0]='$';
for(i=0;i<n;i++)
{
for(l=0;l<n;l++)
{
for(j=0;j<pro[i].n;j++)
{
ptr=strchr(pro[l].rhs[j],temp2[i]);
if(ptr)
{
p=ptr-pro[l].rhs[j];
if(pro[l].rhs[j][p+1]>=65 && pro[l].rhs[j][p+1]<=90)
{
for(o=0;o<n;o++)
if(pro[o].lhs[0]==pro[l].rhs[j][p+1])
strcat(pro[i].fol,pro[o].ft);
}
else if(pro[l].rhs[j][p+1]=='\0')
{
temp=pro[l].lhs[0];
if(pro[l].rhs[j][p]==temp)
continue;
if(temp=='S')
strcat(pro[i].fol,"$");
findfol();
}
else
pro[i].fol[strlen(pro[i].fol)]=pro[l].rhs[j][p+1];
}
}
}
}
11:39 AM
2/25/2022


(27) WhatsApp
Downloads
Ex 5: Computation of FIRST
Online C++ Compiler - onli
C code to Find Firs
onlinegdb.com/online_c_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Run
Debug
Stop
Share
Save
{ } Beautify
main.c
tab5.txt
S ABE
S a
A p
A t
B Aq
S f
A w

OUTPUT:



RESULT:

The computation of First and Follow in C was compiled, executed and verified successfully.

S. AKANSHA
RA1911029010060 CSE-CN P1

EXPERIMENT-6 CONSTRUCTION OF PREDICTIVE PARSING

AIM:

A program to construct Predictive Parsing.

ALGORITHM:

1. Start the program.
2. Initialize the required variables.
3. Get the number of coordinates and productions from the user.
4. Perform the following:
   for (each production A → α in G) {
   for (each terminal a in FIRST(α))
   add A → α to M[A, a];
   if (ε is in FIRST(α))
   for (each symbol b in FOLLOW(A))
   add A → α to M[A, b];
5. Print the resulting stack.
6. Print if the grammar is accepted or not.
7. Exit the program.

CODE:

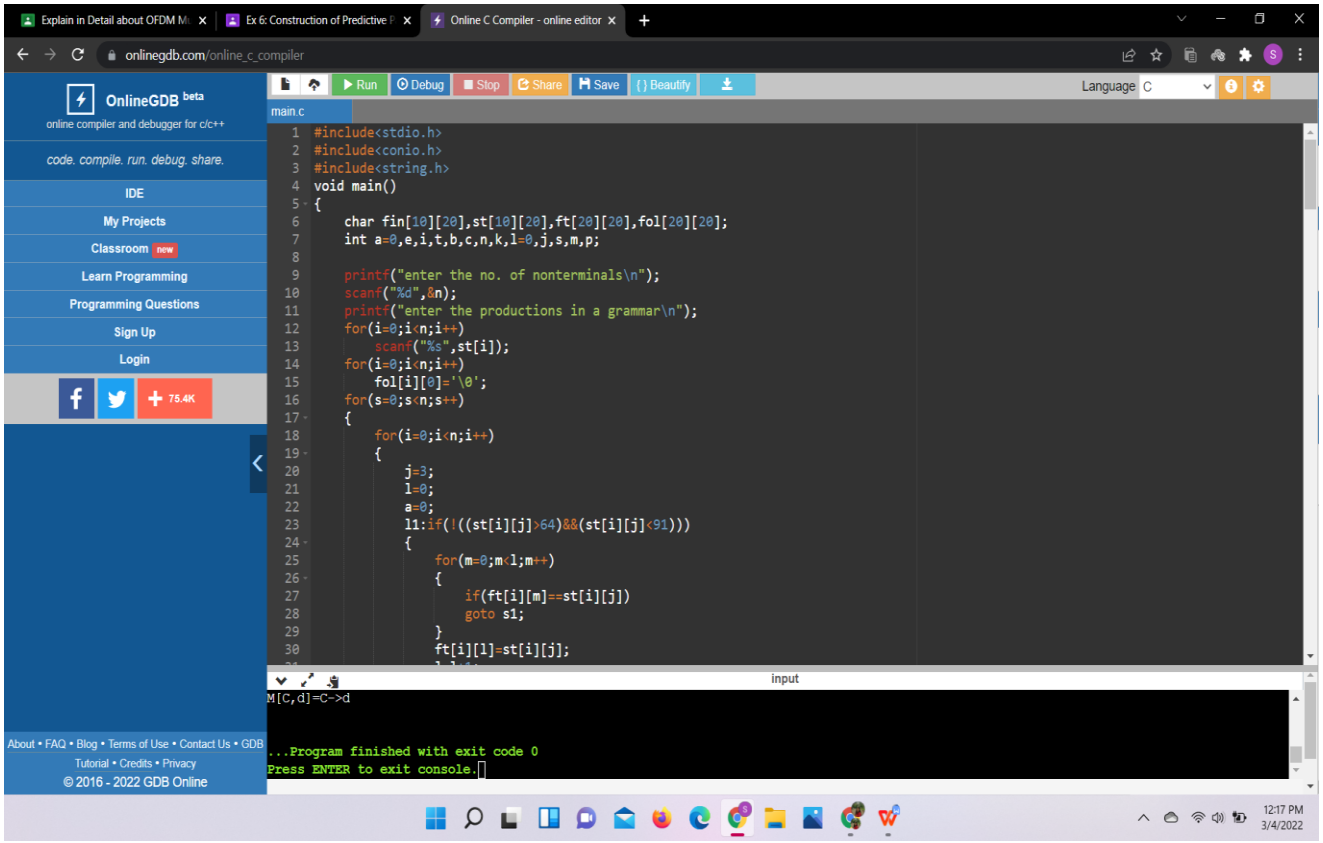
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
   char fin[10][20],st[10][20],ft[20][20],fol[20][20];
   int a=0,e,i,t,b,c,n,k,l=0,j,s,m,p;

   printf("enter the no. of nonterminals\n");
   scanf("%d",&n);
   printf("enter the productions in a grammar\n");
   for(i=0;i<n;i++)
      scanf("%s",st[i]);
   for(i=0;i<n;i++)
      fol[i][0]='\0';
   for(s=0;s<n;s++)
   {
      for(i=0;i<n;i++)
      {
         j=3;
         l=0;
         a=0;
         l1:if(!((st[i][j]>64)&&(st[i][j]<91)))
         {
```

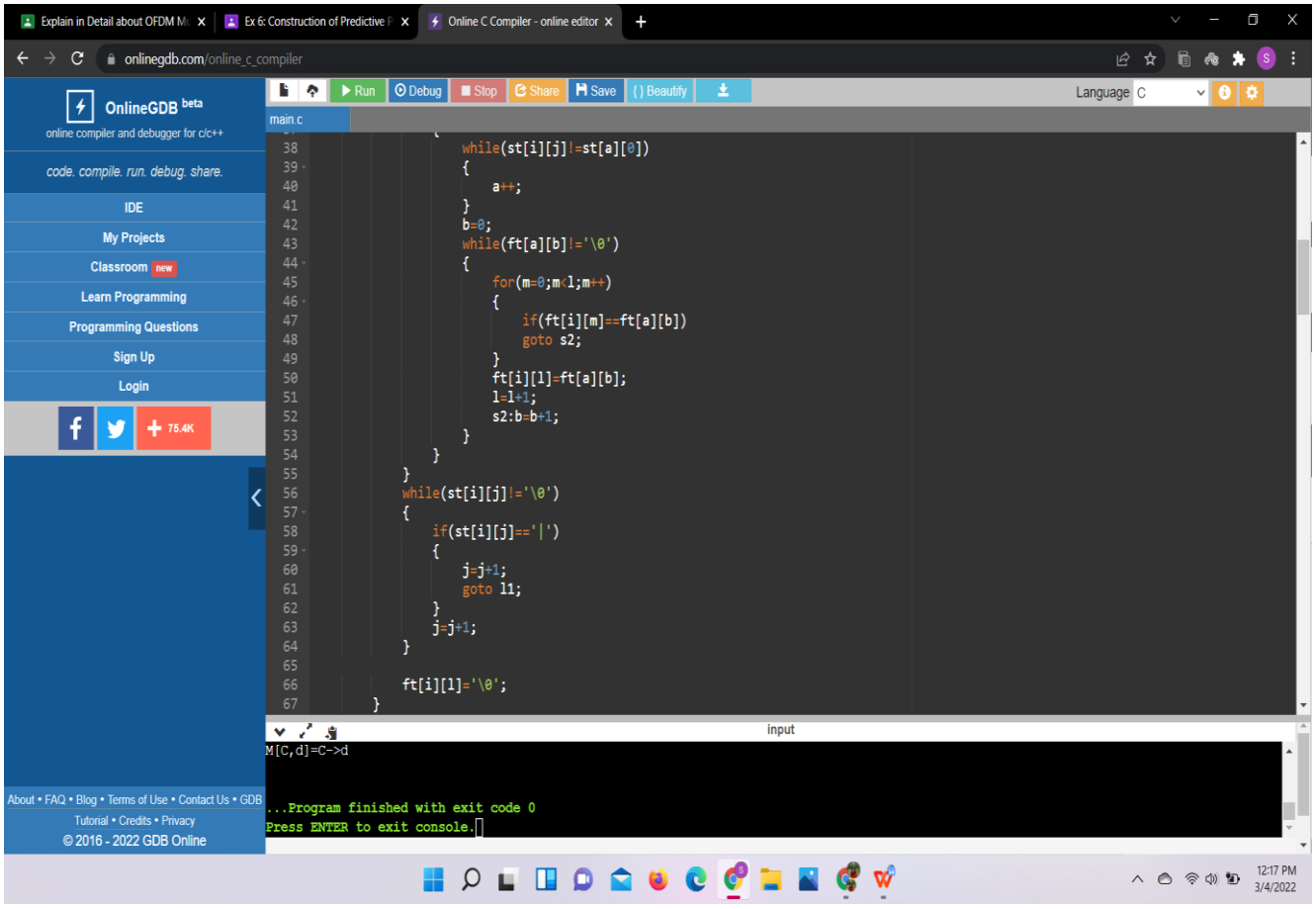
```
            for(m=0;m<l;m++)
            {
               if(ft[i][m]==st[i][j])
               goto s1;
            }
            ft[i][l]=st[i][j];
            l=l+1;
            s1:j=j+1;
         }
         else
         {
            if(s>0)
            {
               while(st[i][j]!=st[a][0])
               {
                  a++;
               }
               b=0;
               while(ft[a][b]!='\0')
               {
                  for(m=0;m<l;m++)
                  {
                     if(ft[i][m]==ft[a][b])
                     goto s2;
                  }
                  ft[i][l]=ft[a][b];
                  l=l+1;
                  s2:b=b+1;
               }
            }
         }
         while(st[i][j]!='\0')
         {
            if(st[i][j]=='|')
            {
               j=j+1;
               goto l1;
            }
            j=j+1;
         }

         ft[i][l]='\0';
      }
   }
   printf("first \n");
   for(i=0;i<n;i++)
      printf("FIRS[%c]=%s\n",st[i][0],ft[i]);
   fol[0][0]='$';
   for(i=0;i<n;i++)
   {
```

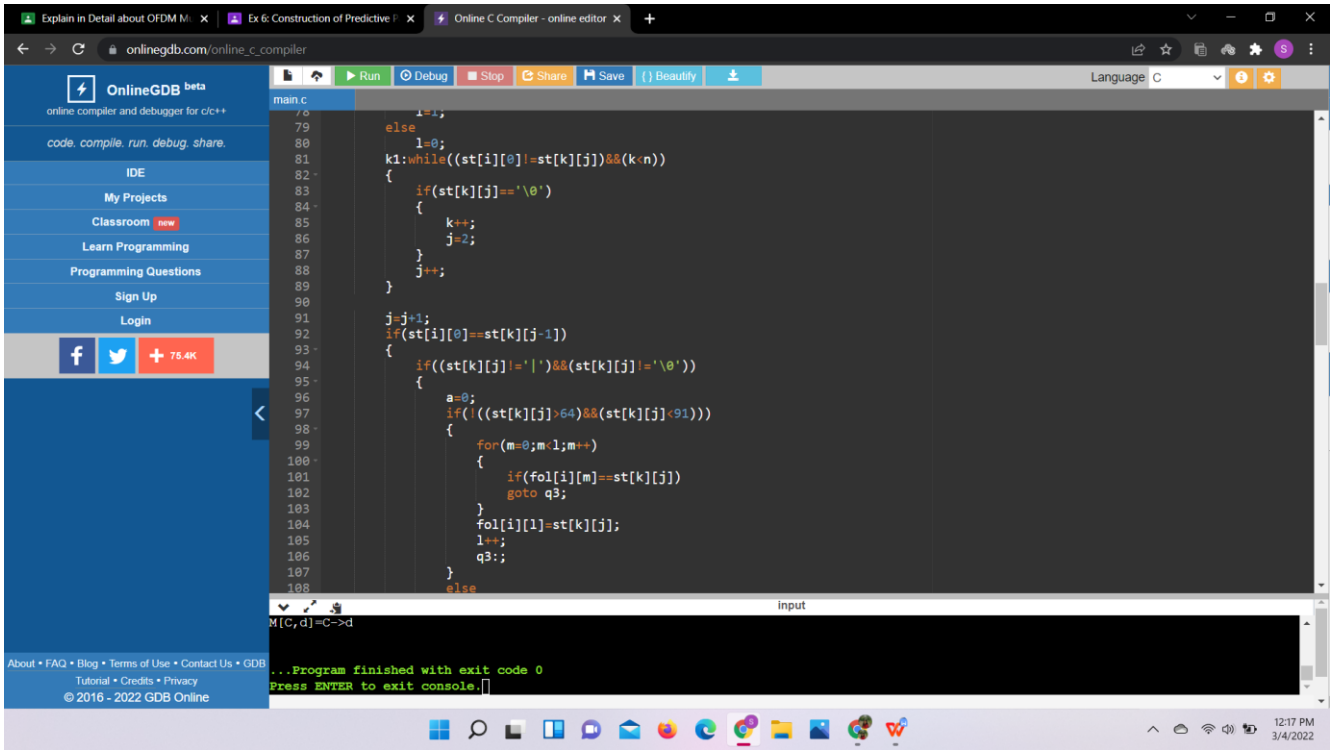
```
k=0;
j=3;
if(i==0)
   l=1;
else
   l=0;
k1:while((st[i][0]!=st[k][j])&&(k<n))
{
   if(st[k][j]=='\0')
   {
      k++;
      j=2;
   }
   j++;
}

j=j+1;
if(st[i][0]==st[k][j-1])
{
   if((st[k][j]!='|')&&(st[k][j]!='\0'))
   {
      a=0;
      if(!((st[k][j]>64)&&(st[k][j]<91)))
      {
         for(m=0;m<l;m++)
         {
            if(fol[i][m]==st[k][j])
            goto q3;
         }
         fol[i][l]=st[k][j];
         l++;
         q3:;
      }
      else
      {
         while(st[k][j]!=st[a][0])
         {
            a++;
         }
         p=0;
         while(ft[a][p]!='\0')
         {
            if(ft[a][p]!='@')
            {
               for(m=0;m<l;m++)
               {
                  if(fol[i][m]==ft[a][p])
                  goto q2;
               }
               fol[i][l]=ft[a][p];
```

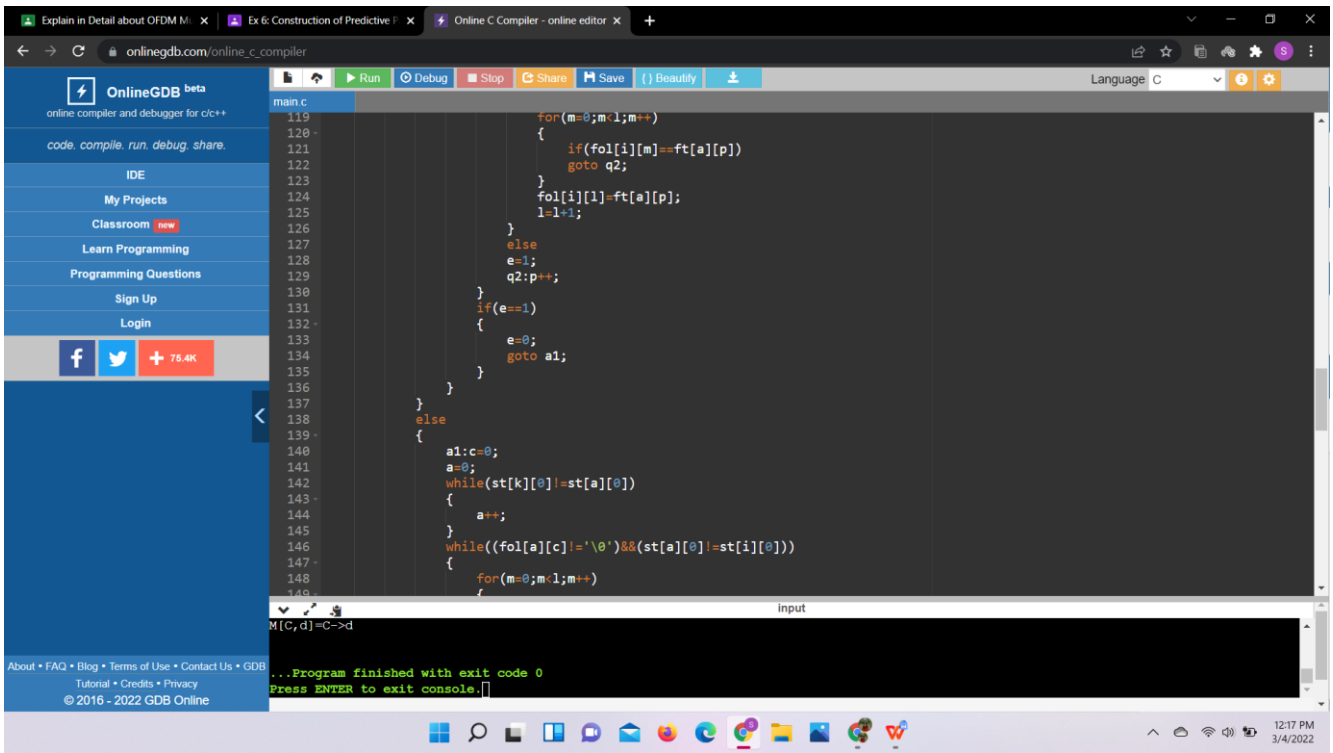
```
                    l=l+1;
                }
                else
                e=1;
                q2:p++;
            }
            if(e==1)
            {
                e=0;
                goto a1;
            }
        }
    }
    else
    {
        a1:c=0;
        a=0;
        while(st[k][0]!=st[a][0])
        {
            a++;
        }
        while((fol[a][c]!='\0')&&(st[a][0]!=st[i][0]))
        {
            for(m=0;m<l;m++)
            {
                if(fol[i][m]==fol[a][c])
                goto q1;
            }
            fol[i][l]=fol[a][c];
            l++;
            q1:c++;
        }
    }
    goto k1;
  }
  fol[i][l]='\0';
}
printf("follow \n");
for(i=0;i<n;i++)
  printf("FOLLOW[%c]=%s\n",st[i][0],fol[i]);
printf("\n");
s=0;
for(i=0;i<n;i++)
{
  j=3;
  while(st[i][j]!='\0')
  {
    if((st[i][j-1]=='|')||(j==3))
    {
        for(p=0;p<=2;p++)
```

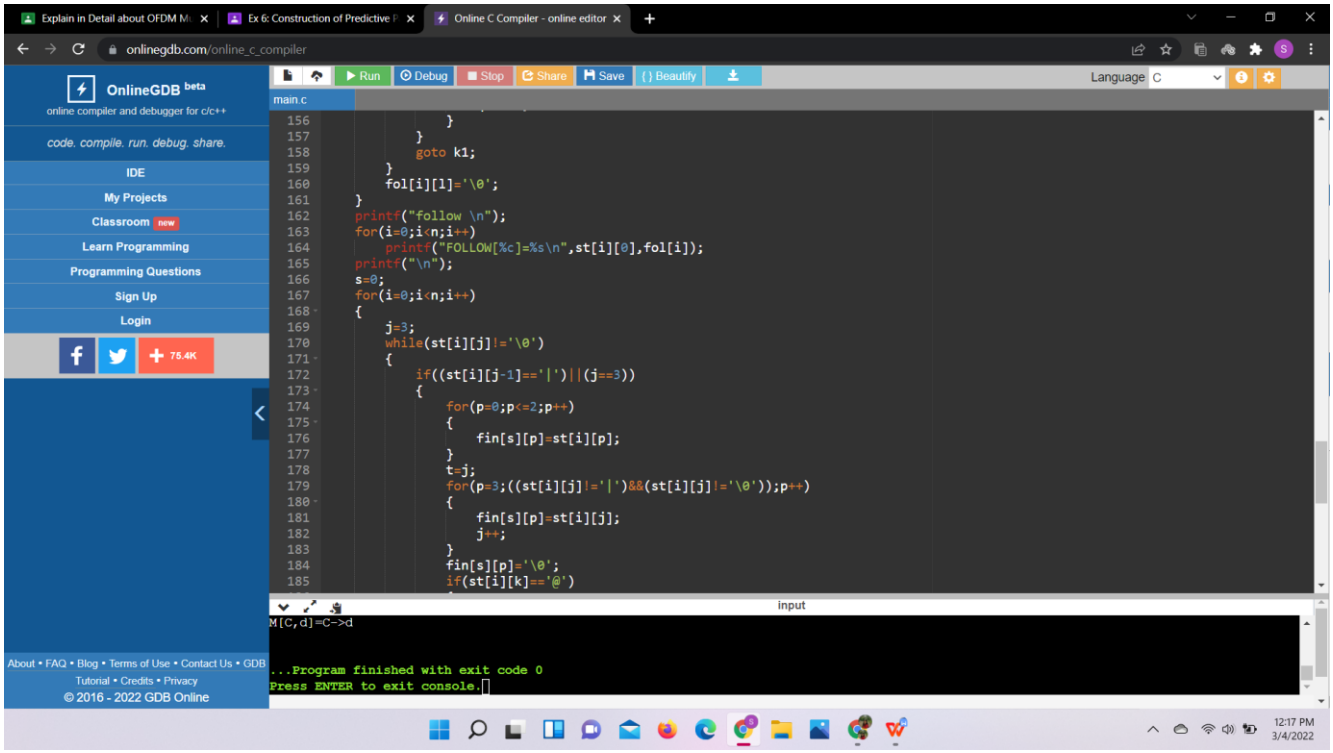
```
            {
               fin[s][p]=st[i][p];
            }
            t=j;
            for(p=3;((st[i][j]!='|')&&(st[i][j]!='\0'));p++)
            {
               fin[s][p]=st[i][j];
               j++;
            }
            fin[s][p]='\0';
            if(st[i][k]=='@')
            {
               b=0;
               a=0;
               while(st[a][0]!=st[i][0])
               {
                  a++;
               }
               while(fol[a][b]!='\0')
               {
                  printf("M[%c,%c]=%s\n",st[i][0],fol[a][b],fin[s]);
                  b++;
               }
            }
            else if(!((st[i][t]>64)&&(st[i][t]<91)))
               printf("M[%c,%c]=%s\n",st[i][0],st[i][t],fin[s]);
            else
            {
               b=0;
               a=0;
               while(st[a][0]!=st[i][3])
               {
                  a++;
               }
               while(ft[a][b]!='\0')
               {
                  printf("M[%c,%c]=%s\n",st[i][0],ft[a][b],fin[s]);
                  b++;
               }
            }
            s++;
         }
         if(st[i][j]=='|')
         j++;
      }
   }

}
```

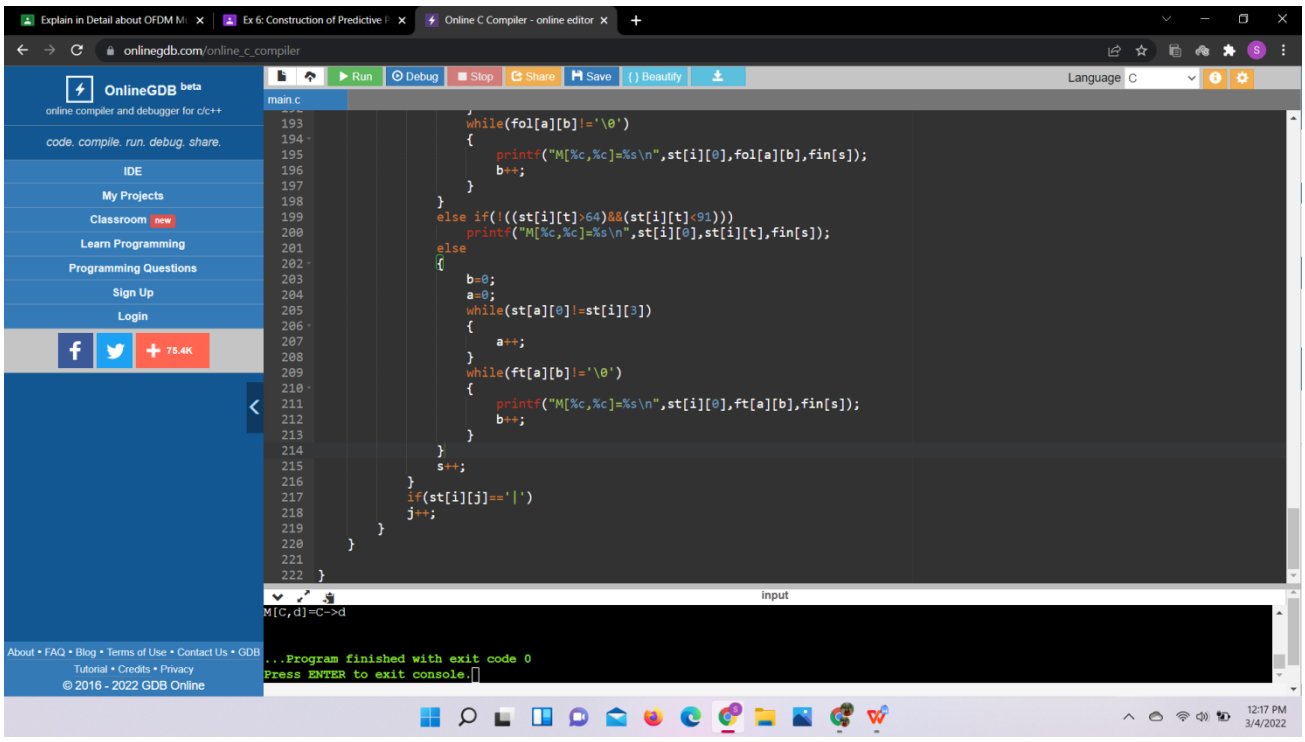

Explain in Detail about OFDM M
Ex 6: Construction of Predictive P
Online C Compiler - online editor
onlinegdb.com/online_c_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
75.4K
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
char fin[10][20],st[10][20],ft[20][20],fol[20][20];
int a=0,e,i,t,b,c,n,k,l=0,j,s,m,p;
printf("enter the no. of nonterminals\n");
scanf("%d",&n);
printf("enter the productions in a grammar\n");
for(i=0;i<n;i++)
scanf("%s",st[i]);
for(i=0;i<n;i++)
fol[i][0]='\0';
for(s=0;s<n;s++)
{
for(i=0;i<n;i++)
{
j=3;
l=0;
a=0;
l1:if(!((st[i][j]>64)&&(st[i][j]<91)))
{
for(m=0;m<l;m++)
{
if(ft[i][m]==st[i][j])
goto s1;
}
ft[i][l]=st[i][j];
input
M[C,d]=C->d
...Program finished with exit code 0
Press ENTER to exit console.
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2022 GDB Online
12:17 PM
3/4/2022


Explain in Detail about OFDM M
Ex 6: Construction of Predictive P
Online C Compiler - online editor
onlinegdb.com/online_c_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
75.4K
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
while(st[i][j]!=st[a][0])
{
a++;
}
b=0;
while(ft[a][b]!='\0')
{
for(m=0;m<l;m++)
{
if(ft[i][m]==ft[a][b])
goto s2;
}
ft[i][l]=ft[a][b];
l=l+1;
s2:b=b+1;
}
}
}
while(st[i][j]!='\0')
{
if(st[i][j]=='|')
{
j=j+1;
goto l1;
}
j=j+1;
}
ft[i][l]='\0';
}
input
M[C,d]=C->d
...Program finished with exit code 0
Press ENTER to exit console.
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2022 GDB Online
12:17 PM
3/4/2022


Explain in Detail about OFDM M
Ex 6: Construction of Predictive P
Online C Compiler - online editor
onlinegdb.com/online_c_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
75.4K
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
else
l=0;
k1:while((st[i][0]!=st[k][j])&&(k<n))
{
if(st[k][j]=='\0')
{
k++;
j=2;
}
j++;
}
j=j+1;
if(st[i][0]==st[k][j-1])
{
if((st[k][j]!='|')&&(st[k][j]!='\0'))
{
a=0;
if(!((st[k][j]>64)&&(st[k][j]<91)))
{
for(m=0;m<l;m++)
{
if(fol[i][m]==st[k][j])
goto q3;
}
fol[i][l]=st[k][j];
l++;
q3:;
}
else
input
M[C,d]=C->d
...Program finished with exit code 0
Press ENTER to exit console.
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2022 GDB Online
12:17 PM
3/4/2022


Explain in Detail about OFDM M
Ex 6: Construction of Predictive P
Online C Compiler - online editor
onlinegdb.com/online_c_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
75.4K
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
for(m=0;m<l;m++)
{
if(fol[i][m]==ft[a][p])
goto q2;
}
fol[i][l]=ft[a][p];
l=l+1;
}
else
e=1;
q2:p++;
}
if(e==1)
{
e=0;
goto a1;
}
}
}
else
{
a1:c=0;
a=0;
while(st[k][0]!=st[a][0])
{
a++;
}
while((fol[a][c]!='\0')&&(st[a][0]!=st[i][0]))
{
for(m=0;m<l;m++)
input
M[C,d]=C->d
...Program finished with exit code 0
Press ENTER to exit console.
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2022 GDB Online
12:17 PM
3/4/2022

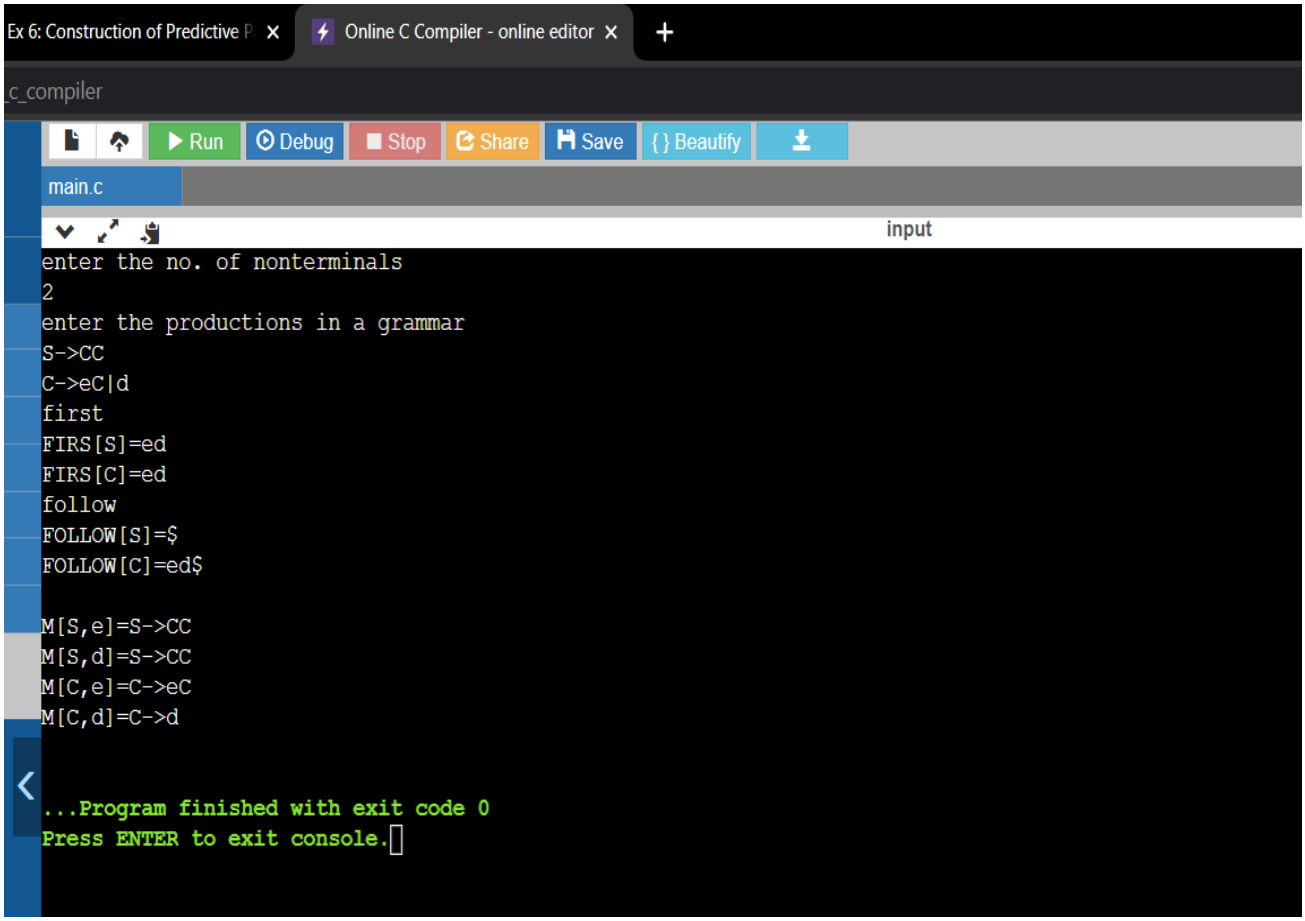

Explain in Detail about OFDM M
Ex 6: Construction of Predictive P
Online C Compiler - online editor
onlinegdb.com/online_c_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
75.4K
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
156 }
157 }
158 goto k1;
159 }
160 fol[i][l]='\0';
161 }
162 printf("follow \n");
163 for(i=0;i<n;i++)
164 printf("FOLLOW[%c]=%s\n",st[i][0],fol[i]);
165 printf("\n");
166 s=0;
167 for(i=0;i<n;i++)
168 {
169 j=3;
170 while(st[i][j]!='\0')
171 {
172 if((st[i][j-1]=='|')||(j==3))
173 {
174 for(p=0;p<=2;p++)
175 {
176 fin[s][p]=st[i][p];
177 }
178 t=j;
179 for(p=3;((st[i][j]!='|')&&(st[i][j]!='\0'));p++)
180 {
181 fin[s][p]=st[i][j];
182 j++;
183 }
184 fin[s][p]='\0';
185 if(st[i][k]=='@')
input
M[C,d]=C->d
...Program finished with exit code 0
Press ENTER to exit console.
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2022 GDB Online
12:17 PM
3/4/2022


Explain in Detail about OFDM M
Ex 6: Construction of Predictive P
Online C Compiler - online editor
onlinegdb.com/online_c_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
75.4K
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
193 while(fol[a][b]!='\0')
194 {
195 printf("M[%c,%c]=%s\n",st[i][0],fol[a][b],fin[s]);
196 b++;
197 }
198 }
199 else if(!((st[i][t]>64)&&(st[i][t]<91)))
200 printf("M[%c,%c]=%s\n",st[i][0],st[i][t],fin[s]);
201 else
202 {
203 b=0;
204 a=0;
205 while(st[a][0]!=st[i][3])
206 {
207 a++;
208 }
209 while(ft[a][b]!='\0')
210 {
211 printf("M[%c,%c]=%s\n",st[i][0],ft[a][b],fin[s]);
212 b++;
213 }
214 }
215 s++;
216 }
217 if(st[i][j]=='|')
218 j++;
219 }
220 }
221
222 }
input
M[C,d]=C->d
...Program finished with exit code 0
Press ENTER to exit console.
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2022 GDB Online
12:17 PM
3/4/2022

OUTPUT:

Enter the no. of nonterminals
2
Enter the productions in a grammar
S->CC
C->eC | d
first
FIRS[S] = ed
FIRS[C] = ed
follow
FOLLOW[S] =$
FOLLOW[C] =ed$

M [S , e] =S->CC
M [S , d] =S->CC
M [C , e] =C->eC
M [C , d] =C->d



RESULT:

The program was successfully compiled and executed.

T. AKANSHA
RA1911029010060 CSE-CN P1

EXPERIMENT-7 Implementation of Shift Reduce Parsing

AIM:

A program to implement shift reduce parsing.

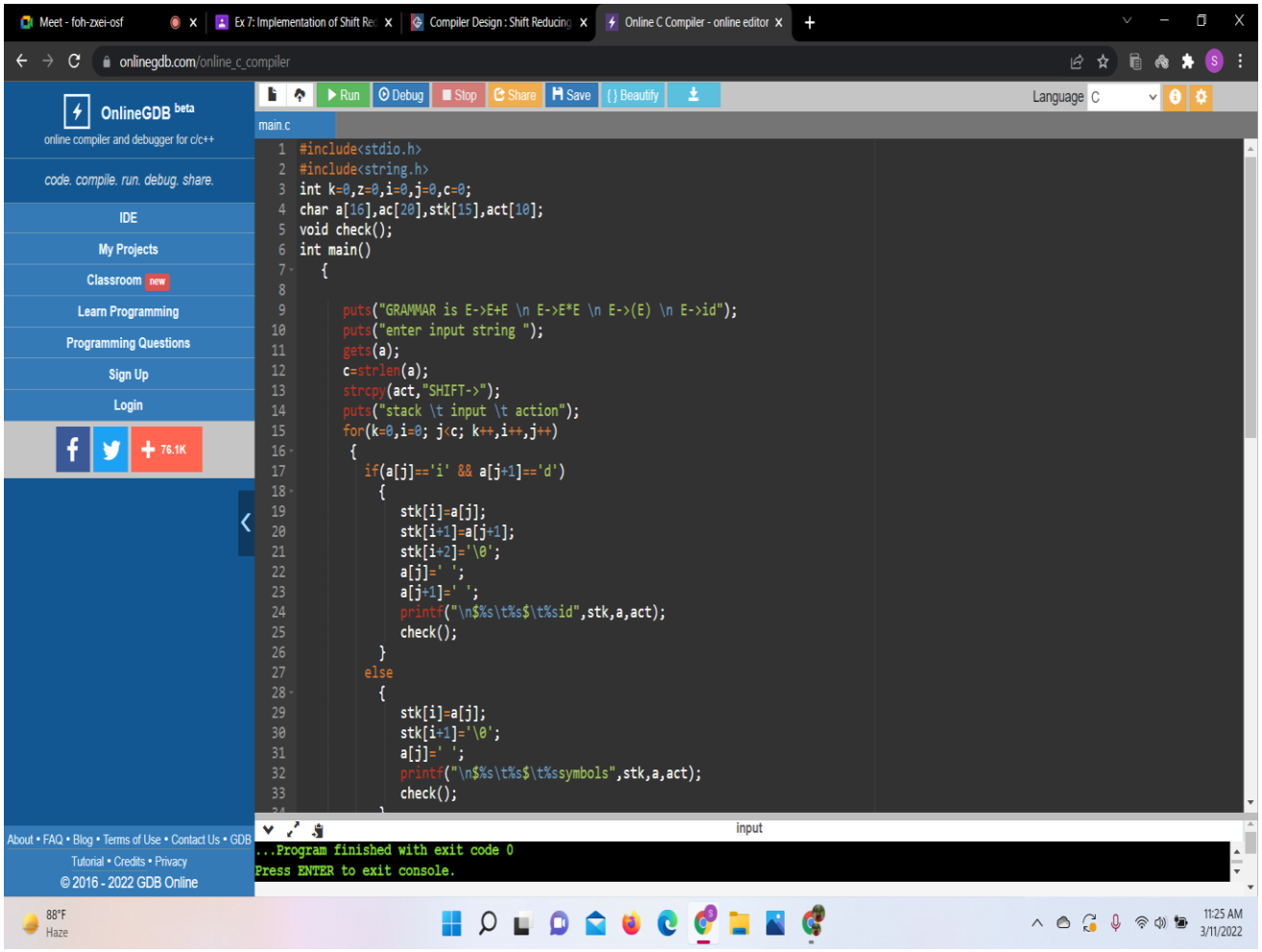
ALGORITHM:

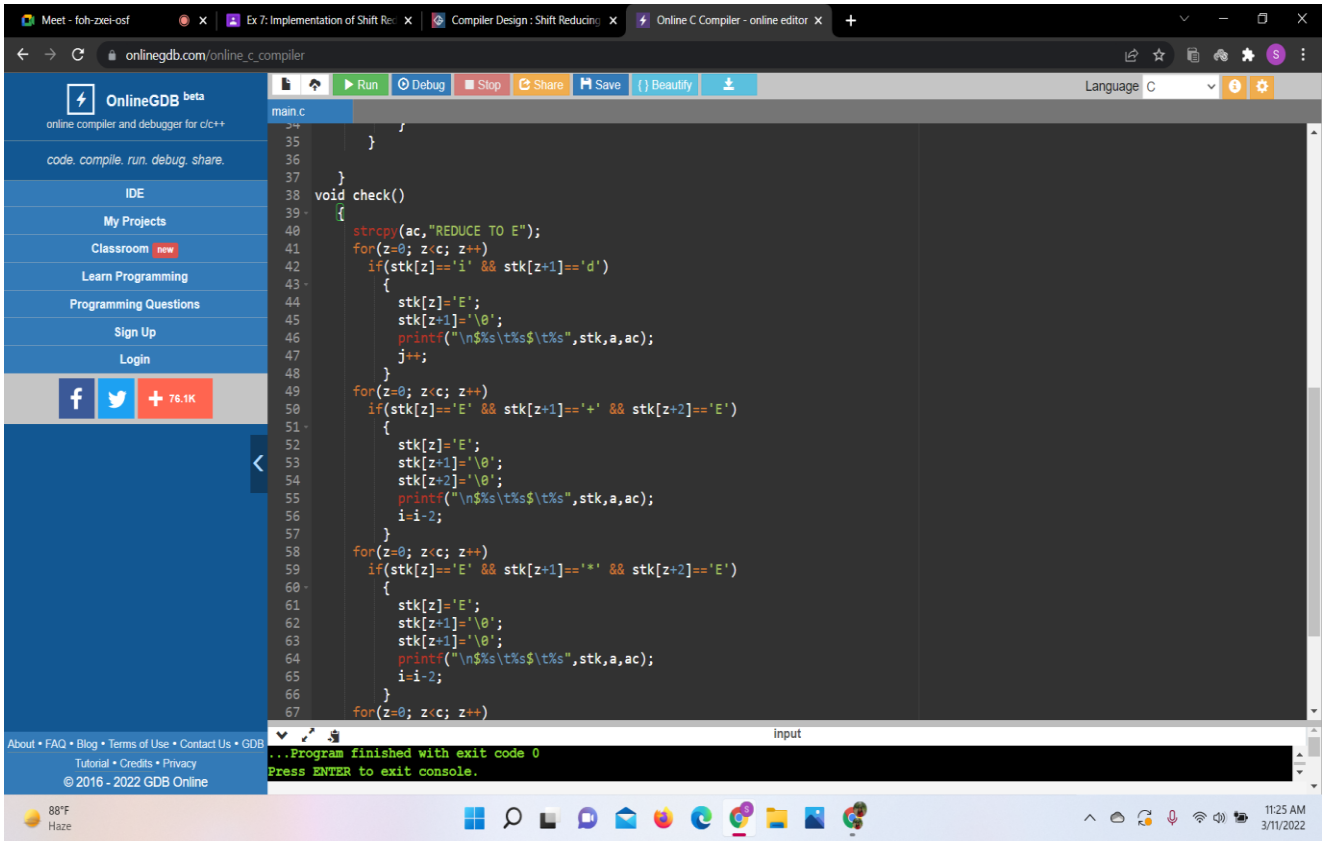
1. Get the input expression and store it in the input buffer.
2. Read the data from the input buffer one at the time.
3. Using stack and push & pop operation shift and reduce symbols with respect to production rules available.
4. Continue the process till symbol shift and production rule reduce reaches the start symbol.
5. Display the Stack Implementation table with corresponding Stack actions with input symbols.

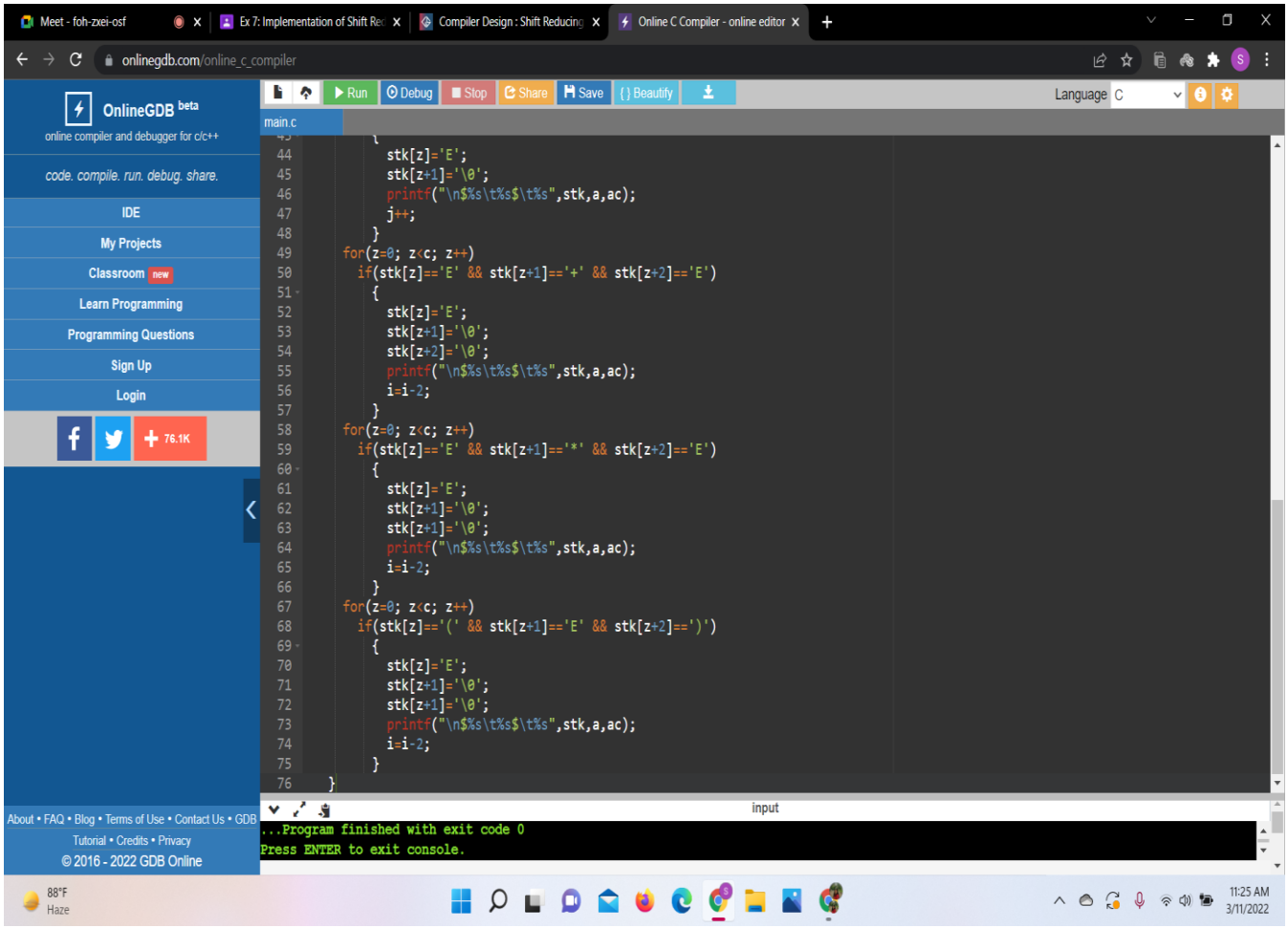
CODE:

```
#include<stdio.h>
#include<string.h>
int k=0,z=0,i=0,j=0,c=0;
char a[16],ac[20],stk[15],act[10];
void check();
int main()
  {

    puts("GRAMMAR is E->E+E \n E->E*E \n E->(E) \n E->id");
    puts("enter input string ");
    gets(a);
    c=strlen(a);
    strcpy(act,"SHIFT->");
    puts("stack \t input \t action");
    for(k=0,i=0; j<c; k++,i++,j++)
     {
      if(a[j]=='i' && a[j+1]=='d')
        {
          stk[i]=a[j];
          stk[i+1]=a[j+1];
          stk[i+2]='\0';
          a[j]=' ';
          a[j+1]=' ';
          printf("\n$%s\t%s$\t%sid",stk,a,act);
          check();
        }
       else
        {
          stk[i]=a[j];
          stk[i+1]='\0';
          a[j]=' ';
          printf("\n$%s\t%s$\t%ssymbols",stk,a,act);
```

```
            check();
          }
       }

   }
void check()
   {
    strcpy(ac,"REDUCE TO E");
    for(z=0; z<c; z++)
      if(stk[z]=='i' && stk[z+1]=='d')
        {
         stk[z]='E';
         stk[z+1]='\0';
         printf("\n$%s\t%s$\t%s",stk,a,ac);
         j++;
        }
    for(z=0; z<c; z++)
      if(stk[z]=='E' && stk[z+1]=='+' && stk[z+2]=='E')
        {
         stk[z]='E';
         stk[z+1]='\0';
         stk[z+2]='\0';
         printf("\n$%s\t%s$\t%s",stk,a,ac);
         i=i-2;
        }
    for(z=0; z<c; z++)
      if(stk[z]=='E' && stk[z+1]=='*' && stk[z+2]=='E')
        {
         stk[z]='E';
         stk[z+1]='\0';
         stk[z+1]='\0';
         printf("\n$%s\t%s$\t%s",stk,a,ac);
         i=i-2;
        }
    for(z=0; z<c; z++)
      if(stk[z]=='(' && stk[z+1]=='E' && stk[z+2]==')')
        {
         stk[z]='E';
         stk[z+1]='\0';
         stk[z+1]='\0';
         printf("\n$%s\t%s$\t%s",stk,a,ac);
         i=i-2;
        }
   }
```


Meet - foh-zxei-osf
Ex 7: Implementation of Shift Re
Compiler Design : Shift Reducing
Online C Compiler - online editor
onlinegdb.com/online_c_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
76.1K
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
#include<stdio.h>
#include<string.h>
int k=0,z=0,i=0,j=0,c=0;
char a[16],ac[20],stk[15],act[10];
void check();
int main()
{
puts("GRAMMAR is E->E+E \n E->E*E \n E->(E) \n E->id");
puts("enter input string ");
gets(a);
c=strlen(a);
strcpy(act,"SHIFT->");
puts("stack \t input \t action");
for(k=0,i=0; j<c; k++,i++,j++)
{
if(a[j]=='i' && a[j+1]=='d')
{
stk[i]=a[j];
stk[i+1]=a[j+1];
stk[i+2]='\0';
a[j]=' ';
a[j+1]=' ';
printf("\n$%s\t%s$\t%sid",stk,a,act);
check();
}
else
{
stk[i]=a[j];
stk[i+1]='\0';
a[j]=' ';
printf("\n$%s\t%s$\t%ssymbols",stk,a,act);
check();
input
...Program finished with exit code 0
Press ENTER to exit console.
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2022 GDB Online
88°F
Haze
11:25 AM
3/11/2022

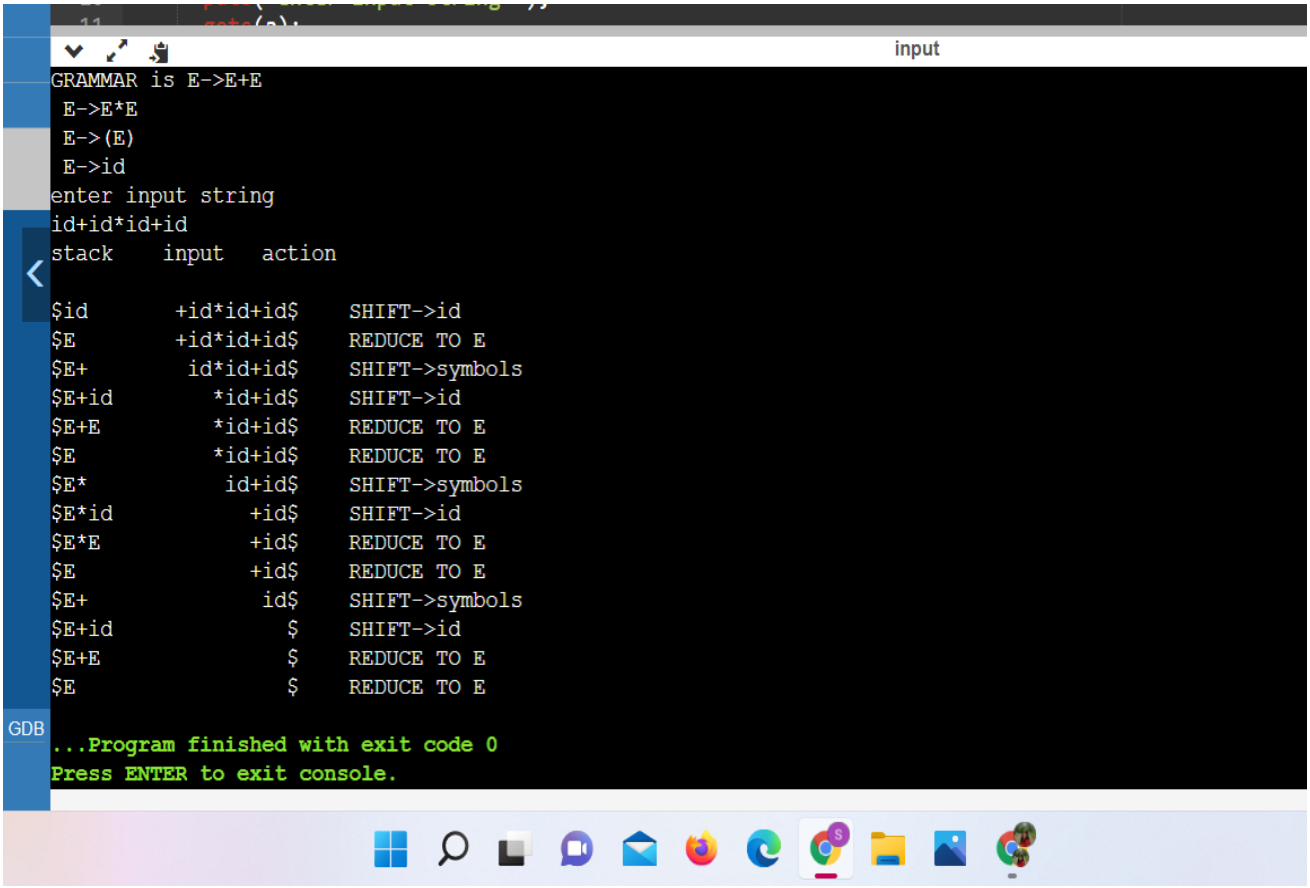

Meet - foh-zxei-osf
Ex 7: Implementation of Shift Re
Compiler Design : Shift Reducing
Online C Compiler - online editor
onlinegdb.com/online_c_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
76.1K
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
}
}
}
void check()
{
strcpy(ac,"REDUCE TO E");
for(z=0; z<c; z++)
if(stk[z]=='i' && stk[z+1]=='d')
{
stk[z]='E';
stk[z+1]='\0';
printf("\n$%s\t%s$\t%s",stk,a,ac);
j++;
}
for(z=0; z<c; z++)
if(stk[z]=='E' && stk[z+1]=='+' && stk[z+2]=='E')
{
stk[z]='E';
stk[z+1]='\0';
stk[z+2]='\0';
printf("\n$%s\t%s$\t%s",stk,a,ac);
i=i-2;
}
for(z=0; z<c; z++)
if(stk[z]=='E' && stk[z+1]=='*' && stk[z+2]=='E')
{
stk[z]='E';
stk[z+1]='\0';
stk[z+1]='\0';
printf("\n$%s\t%s$\t%s",stk,a,ac);
i=i-2;
}
for(z=0; z<c; z++)
input
...Program finished with exit code 0
Press ENTER to exit console.
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2022 GDB Online
88°F
Haze
11:25 AM
3/11/2022



OUTPUT:

GRAMMAR is E->E+E
E->E*E
E->(E)
E->id
enter input string
id+id*id+id
stack input action

$id +id*id+id$ SHIFT->id
$E +id*id+id$ REDUCE TO E
$E+ id*id+id$ SHIFT->symbols
$E+id *id+id$ SHIFT->id
$E+E *id+id$ REDUCE TO E
$E *id+id$ REDUCE TO E
$E* id+id$ SHIFT->symbols
$E*id +id$ SHIFT->id
$E*E +id$ REDUCE TO E
$E +id$ REDUCE TO E
$E+ id$ SHIFT->symbols
$E+id $ SHIFT->id
$E+E $ REDUCE TO E
$E $ REDUCE TO E



RESULT:

The program was successfully compiled and executed.

S. AKANSHA

RA191102901006

0 CSE-CN P1

EXP 8: Construction of Leading and Trailing

Aim:

To write a program for implementing leading and

trailing.

Algorithm:

1. For Leading, check for the first non-terminal.
2. If found, print it.
3. Look for next production for the same non-terminal.
4. If not found, recursively call the procedure for the single non-terminal present before the comma orEnd Of Production String.
5. Include it's results in the result of this non-terminal.
6. For trailing, we compute same as leading but we start from the end of the production to the beginning.
7. Stop

Main.c:

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int vars,terms,i,j,k,m,rep,count,temp=-1;
char var[10],term[10],lead[10][10],trail[10][10];
struct grammar
{
        int prodno;
```

```
        char lhs,rhs[20][20];
}gram[50];
void get()
{
        cout<<"\n---------- LEADING AND TRAILING ---------- \n";
        cout<<"\nEnter the no. of variables : ";
        cin>>vars;
        cout<<"\nEnter the variables : \n";
        for(i=0;i<vars;i++)
        {
                cin>>gram[i].lhs;
                var[i]=gram[i].lhs;
        }
        cout<<"\nEnter the no. of terminals : ";
        cin>>terms;
        cout<<"\nEnter the terminals : ";
        for(j=0;j<terms;j++)
                cin>>term[j];
        cout<<"\n---------- PRODUCTION DETAILS ---------- \n";
        for(i=0;i<vars;i++)
        {
                cout<<"\nEnter the no. of production of "<<gram[i].lhs<<":";
                cin>>gram[i].prodno;
                for(j=0;j<gram[i].prodno;j++)
                {
                        cout<<gram[i].lhs<<"->";
                        cin>>gram[i].rhs[j];
                }
        }
}
void leading()
```

```
{
        for(i=0;i<vars;i++)
        {
                for(j=0;j<gram[i].prodno;j++)
                {
                        for(k=0;k<terms;k++)
                        {
                                if(gram[i].rhs[j][0]==term[k])
                                        lead[i][k]=1;
                                else
                                {
                                        if(gram[i].rhs[j][1]==term[k])
                                                lead[i][k]=1;
                                }
                        }
                }
        }

        for(rep=0;rep<vars;rep++)
        {
                for(i=0;i<vars;i++)
                {
                        for(j=0;j<gram[i].prodno;j++)
                        {
                                for(m=1;m<vars;m++)
                                {
                                        if(gram[i].rhs[j][0]==var[m])
                                        {
                                                temp=m;
                                                goto out;
                                        }
                                }
```

```
                        out:
                        for(k=0;k<terms;k++)
                        {
                                if(lead[temp][k]==1)
                                        lead[i][k]=1;
                        }
                }
        }
}
void trailing()
{
        for(i=0;i<vars;i++)
        {
                for(j=0;j<gram[i].prodno;j++)
                {
                        count=0;
                        while(gram[i].rhs[j][count]!='\x0')
                                count++;
                        for(k=0;k<terms;k++)
                        {
                                if(gram[i].rhs[j][count-1]==term[k])
                                        trail[i][k]=1;
                                else
                                {
                                        if(gram[i].rhs[j][count-2]==term[k])
                                                trail[i][k]=1;
                                }
                        }
                }
        }
```

```
	for(rep=0;rep<vars;rep++)
	{
		for(i=0;i<vars;i++)
		{
			for(j=0;j<gram[i].prodno;j++)
			{
				count=0;
				while(gram[i].rhs[j][count]!='\x0')
					count++;
				for(m=1;m<vars;m++)
				{
					if(gram[i].rhs[j][count-1]==var[m])
						temp=m;
				}
				for(k=0;k<terms;k++)
				{
					if(trail[temp][k]==1)
						trail[i][k]=1;
				}
			}
		}
	}
}
void display()
{
	for(i=0;i<vars;i++)
	{
		cout<<"\nLEADING("<<gram[i].lhs<<") = ";
		for(j=0;j<terms;j++)
		{
			if(lead[i][j]==1)
```

```
                        cout<<term[j]<<",";
                }
        }
        cout<<endl;
        for(i=0;i<var
        s;i++)
        {
                cout<<"\nTRAILING("<<gram[i].lhs<<"
                ) = "; for(j=0;j<terms;j++)
                {
                        if(trail[i][j]==1)
                                cout<<term[j]<<",";
                }
        }
}
void main()
{
clrscr();
get();
leading();
trailing();
display();
getch();
}
```

Output:

```
------------ LEADING AND TRAILING ---------------

Enter the no. of variables : 3

Enter the variables :
E
T
F

Enter the no. of terminals : 5

Enter the terminals : )
(
*
+
i

------------ PRODUCTION DETAILS -----------------

Enter the no. of production of E:2
E->E+T
E->T

Enter the no. of production of T:2
T->T*F
T->F

Enter the no. of production of F:2
F->(E)
F->i

LEADING(E) = (,*,+,i,
LEADING(T) = (,*,i,
LEADING(F) = (,i,

TRAILING(E) = ),*,+,i,
TRAILING(T) = ),*,i,
TRAILING(F) = ),i,
```

Result:

The program to implementing leading and trailing was completed successfully.

S. AKANSHA
RA1911029010060 CSE CN P1

EXP 9 : Computation of LR(0) Items

Aim:

A program to implement LR(0) items

Algorithm:-

8. Start.
9. Create structure for production with LHS and RHS.
10. Open file and read input from file.
11. Build state 0 from extra grammar Law S' -> S $ that is all start symbol of grammar and one Dot (.) before S symbol.
12. If Dot symbol is before a non-terminal, add grammar laws that this non-terminal is in LeftHand Side of that Law and set Dot in before of first part of Right Hand Side.
13. If state exists (a state with this Laws and same Dot position), use that instead.
14. Now find set of terminals and non-terminals in which Dot exist in before.
15. If step 7 Set is non-empty go to 9, else go to 10.
16. For each terminal/non-terminal in set step 7 create new state by using all grammar law thatDot position is before of that terminal/non-terminal in reference state by increasing Dot point to next part in Right Hand Side of that laws.
17. Go to step 5.
18. End of state building.
19. Display the output.
20. End.

Program:

```
#include<iostream.h>
#include<conio.h>
#include<string.h>

char prod[20][20],listofvar[26]="ABCDEFGHIJKLMNOPQR";
int novar=1,i=0,j=0,k=0,n=0,m=0,arr[30];
```

```
int noitem=0;
struct Grammar
{
        char lhs; char rhs[8];
}g[20],item[20],clos[20][10];


int isvariable(char variable)
{ for(int i=0;i<novar;i++)
        if(g[i].lhs==variable) return
        i+1;
        return 0;
}
void findclosure(int z, char a)
{ int n=0,i=0,j=0,k=0,l=0;
        for(i=0;i<arr[z];i++)
        {
                for(j=0;j<strlen(clos[z][i].rhs);j++)
                {
                        if(clos[z][i].rhs[j]=='.' && clos[z][i].rhs[j+1]==a)
                        {
                                clos[noitem][n].lhs=clos[z][i].lhs;
                                strcpy(clos[noitem][n].rhs,clos[z][i].rhs); char
                                temp=clos[noitem][n].rhs[j];
                                clos[noitem][n].rhs[j]=clos[noitem][n].rhs[j+1]
                                ; clos[noitem][n].rhs[j+1]=temp; n=n+1;
                        }
                }
        }
        for(i=0;i<n;i++)
        {
                for(j=0;j<strlen(clos[noitem][i].rhs);j++)
                {
```

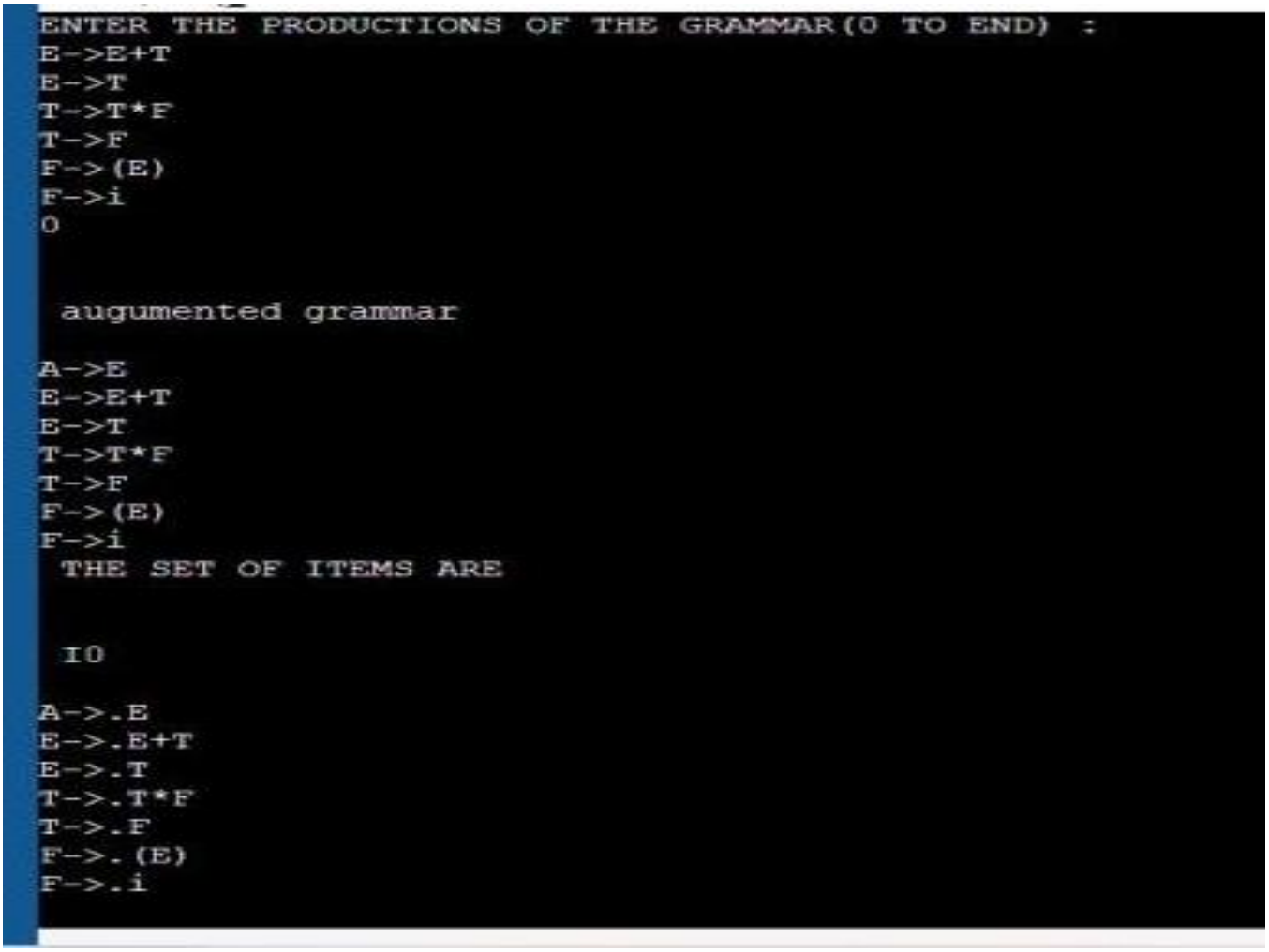
```
                if(clos[noitem][i].rhs[j]=='.' && isvariable(clos[noitem][i].rhs[j+1])>0)
                { for(k=0;k<novar;k++)
                        {
                                if(clos[noitem][i].rhs[j+1]==clos[0][k].lhs)
                                { for(l=0;l<n;l++) if(clos[noitem][l].lhs==clos[0][k].lhs
                                                &&
strcmp(clos[noitem][l].rhs,clos[0][k].rhs)==0)
                                                        break;

                                        if(l==n)
                                        {
                                                clos[noitem][n].lhs=clos[0][k].lhs;
                                        strcpy(clos[noitem][n].rhs,clos[0][k].rhs);
                                                n=n+1;
                                        }
                                }
                        }
                } }
        arr[noitem]=n; int
        flag=0;
        for(i=0;i<noitem;i++)
        {
                if(arr[i]==n)
                {
                        for(j=0;j<arr[i];j++)
                        {
                                int c=0;
                                for(k=0;k<arr[i];k++)
                                        if(clos[noitem][k].lhs==clos[i][k].lhs                   &&
strcmp(clos[noitem][k].rhs,clos[i][k].rhs)==0)
```

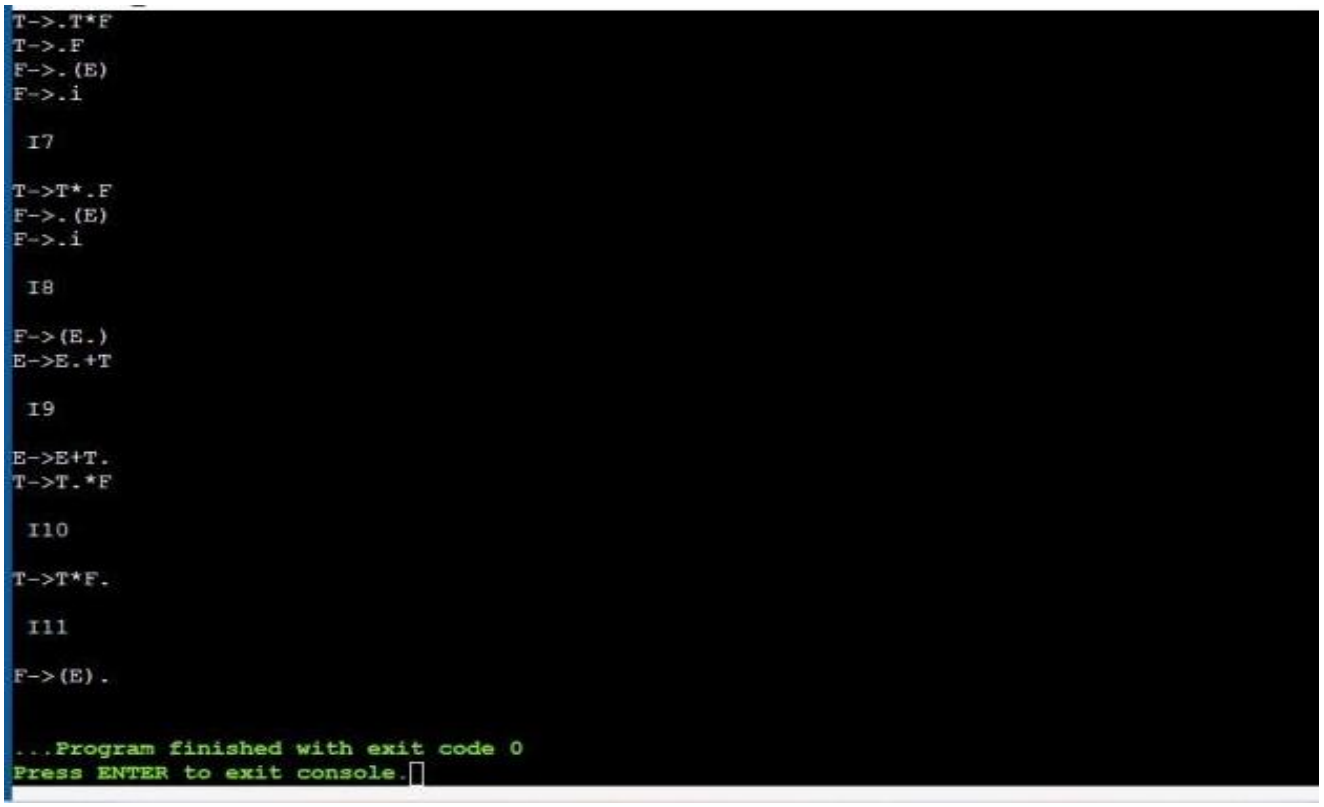
```
                                        c=c+1;
                                if(c==arr[i])
                                { flag=1; goto
                                        exit;
                                }
                        }
                }
        }
        exit:; if(flag==0)
        arr[noitem++]=n;
}

void main()
{
        clrscr();
        cout<<"ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO END) :\n";
        do
        { cin>>prod[i++];
        }while(strcmp(prod[i-1],"0")!=0); for(n=0;n<i-
        1;n++)
        {
                m=0;
                j=novar;
                g[novar++].lhs=prod[n][0];
                for(k=3;k<strlen(prod[n]);k++)
                {
                        if(prod[n][k] != '|')
                        g[j].rhs[m++]=prod[n][k];
                        if(prod[n][k]=='|')
                        {
                                g[j].rhs[m]='\0';
```

```
                    m=0;
                    j=novar;
                    g[novar++].lhs=prod[n][0];
                }
        } } for(i=0;i<26;i++)
if(!isvariable(listofvar[i])) break;
g[0].lhs=listofvar[i]; char temp[2]={g[1].lhs,'\0'};
strcat(g[0].rhs,temp); cout<<"\n\n augumented
grammar \n"; for(i=0;i<novar;i++)
cout<<endl<<g[i].lhs<<"->"<<g[i].rhs<<" ";
getch();
for(i=0;i<novar;i++)
{
        clos[noitem][i].lhs=g[i].lhs;
        strcpy(clos[noitem][i].rhs,g[i].rhs);
        if(strcmp(clos[noitem][i].rhs,"ε")==0)
        strcpy(clos[noitem][i].rhs,".");

        else
        {
                for(int j=strlen(clos[noitem][i].rhs)+1;j>=0;j--)
                        clos[noitem][i].rhs[j]=clos[noitem][i].rhs[j-1];
                clos[noitem][i].rhs[0]='.';
        } }
arr[noitem++]=novar;
for(int z=0;z<noitem;z++)
{
        char list[10]; int
        l=0;
        for(j=0;j<arr[z];j++
        )
        {
```

```
                    for(k=0;k<strlen(clos[z][j].rhs)-1;k++)
                    {
                        if(clos[z][j].rhs[k]=='.')
                        { for(m=0;m<l;m++)
                                if(list[m]==clos[z][j].rhs[k+1])
                                    break;

                            if(m==l)
                                list[l++]=clos[z][j].rhs[k+1];

                        }
                    } } for(int
                x=0;x<l;x++)
                findclosure(z,list[x]);

        }
        cout<<"\n THE SET OF ITEMS ARE \n\n";
        for(z=0;z<noitem;z++)
        { cout<<"\n I"<<z<<"\n\n"; for(j=0;j<arr[z];j++)
                cout<<clos[z][j].lhs<<"->"<<clos[z][j].rhs<<"\n";
                getch();

        }
        getch();
}
```

Output:


ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO END) :
E->E+T
E->T
T->T*F
T->F
F->(E)
F->i
0
augumented grammar
A->E
E->E+T
E->T
T->T*F
T->F
F->(E)
F->i
THE SET OF ITEMS ARE
I0
A->.E
E->.E+T
E->.T
T->.T*F
T->.F
F->.(E)
F->.i



Result:

The program was successfully compiled and run.

EXP 10: Intermediate code generation – Postfix, Prefix

S. AKANSHA
RA1911029010060 CSE-CN P1

Aim:

A program to implement Intermediate code generation – Postfix, Prefix.

Algorithm:

- Declare set of operators.
- Initialize an empty stack.
- To convert INFIX to POSTFIX follow the following steps
- Scan the infix expression from left to right.
- If the scanned character is an operand, output it.
- Else, If the precedence of the scanned operator is greater than the precedence of the operator in the stack(or the stack is empty or the stack contains a '('), push it.
- Else, Pop all the operators from the stack which are greater than or equal to in precedence than that of the scanned operator. After doing that Push the scanned operator to the stack.
- If the scanned character is an '(', push it to the stack.
- If the scanned character is an ')', pop the stack and output it until a '(' is encountered, and discard both the parenthesis.
- Pop and output from the stack until it is not empty.
- To convert INFIX to PREFIX follow the following steps
- First, reverse the infix expression given in the problem.
- Scan the expression from left to right.
- Whenever the operands arrive, print them.
- If the operator arrives and the stack is found to be empty, then simply push the operator into the stack.
- Repeat steps 6 to 9 until the stack is empty.

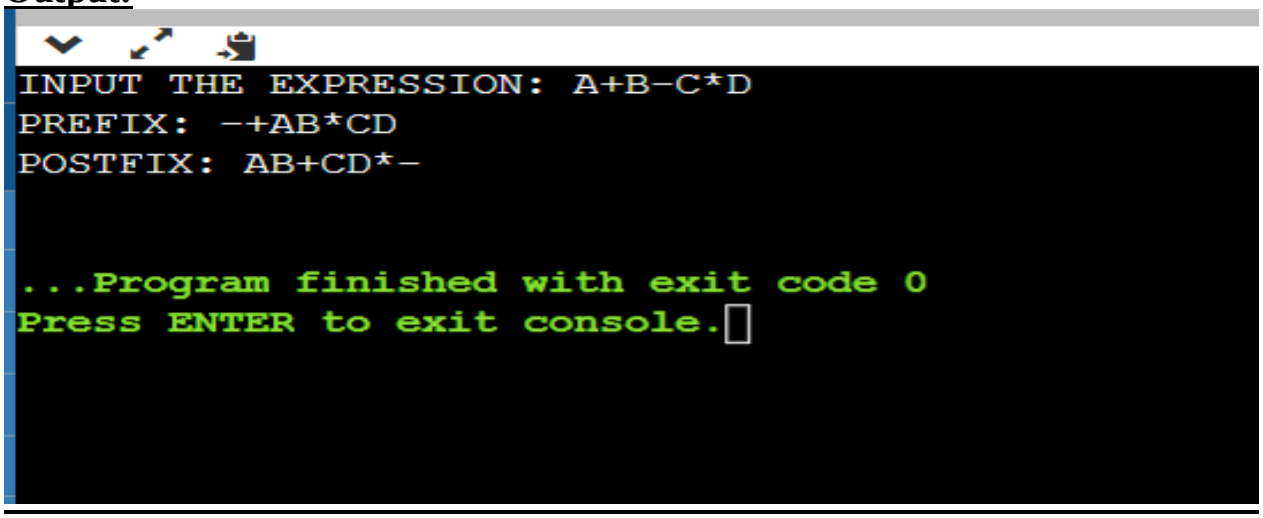
Code:

```
OPERATORS = set(['+', '-', '*', '/', '(', ')'])

PRI = {'+': 1, '-': 1, '*': 2, '/': 2}


### INFIX ===> POSTFIX ###

def infix_to_postfix(formula):
    stack = []  # only pop when the coming op has priority

    output = ''

    for ch in formula:

        if ch not in OPERATORS:

            output += ch

        elif ch == '(':

            stack.append('(')

        elif ch == ')':

            while stack and stack[-1] != '(':
                output += stack.pop()

            stack.pop()  # pop '('

        else:

            while stack and stack[-1] != '(' and PRI[ch] <= PRI[stack[-1]]:
                output += stack.pop()

            stack.append(ch)

            # leftover

    while stack:
        output += stack.pop()

    print(f'POSTFIX: {output}')
```

```
    return output


### INFIX ===> PREFIX ###

def infix_to_prefix(formula):
    op_stack = []

    exp_stack = []

    for ch in formula:

        if not ch in OPERATORS:

            exp_stack.append(ch)

        elif ch == '(':

            op_stack.append(ch)

        elif ch == ')':

            while op_stack[-1] != '(':
                op = op_stack.pop()

                a = exp_stack.pop()

                b = exp_stack.pop()

                exp_stack.append(op + b + a)

            op_stack.pop()  # pop '('

        else:

            while op_stack and op_stack[-1] != '(' and PRI[ch] <= PRI[op_stack[-1]]:
                op = op_stack.pop()

                a = exp_stack.pop()

                b = exp_stack.pop()
```

```
            exp_stack.append(op + b + a)

        op_stack.append(ch)

        # leftover

    while op_stack:
        op = op_stack.pop()

        a = exp_stack.pop()

        b = exp_stack.pop()

        exp_stack.append(op + b + a)

    print(f'PREFIX: {exp_stack[-1]}')

    return exp_stack[-1]

expres = input("INPUT THE EXPRESSION: ")

pre = infix_to_prefix(expres)

pos = infix_to_postfix(expres)
```

Output:



Result:

Thus the program was successfully compiled and run.

S.AKANSHA

RA1911029010060 CSE CN P1

Exp 11: Intermediate code generation – Quadruple, Triple, Indirect triple

Aim:

A program to implement intermediate code generation - Quadruple, Triple, Indirect triple.

Algorithm:

The algorithm takes a sequence of three-address statements as input. For each three address statements of the form a:= b op c perform the various actions. These are as follows:

1. Invoke a function getreg to find out the location L where the result of computation b op c should be stored.
2. Consult the address description for y to determine y'. If the value of y currently in memory and register both then prefer the register y' . If the value of y is not already in L then generate the instruction MOV y' , L to place a copy of y in L.
3. Generate the instruction OP z' , L where z' is used to show the current location of z. if z is in both then prefer a register to a memory location. Update the address descriptor of x to indicate that x is in location L. If x is in L then update its descriptor and remove x from all other descriptors.
4. If the current value of y or z have no next uses or not live on exit from the block or in register then alter the register descriptor to indicate that after execution of x : = y op z those register will no longer contain y or z.

Program:

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

```
void small();
void dove(int i);
int p[5]={0,1,2,3,4},c=1,i,k,l,m,pi;
char sw[5]={'=','-','+','/','*'},j[20],a[5],b[5],ch[2];
void main()
{
   printf("Enter the expression : ");
   scanf("%s",j);
   printf("The Intermediate code is :\n");
   small();
}
void dove(int i)
{
   a[0]=b[0]='\0';
   if(!isdigit(j[i+2])&&!isdigit(j[i-2]))
   {
      a[0]=j[i-1];
      b[0]=j[i+1];
   }
   if(isdigit(j[i+2]))
   {
      a[0]=j[i-1];
      b[0]='t';
      b[1]=j[i+2];
   }
   if(isdigit(j[i-2]))
   {
      b[0]=j[i+1];
      a[0]='t';
      a[1]=j[i-2];
      b[1]='\0';
```

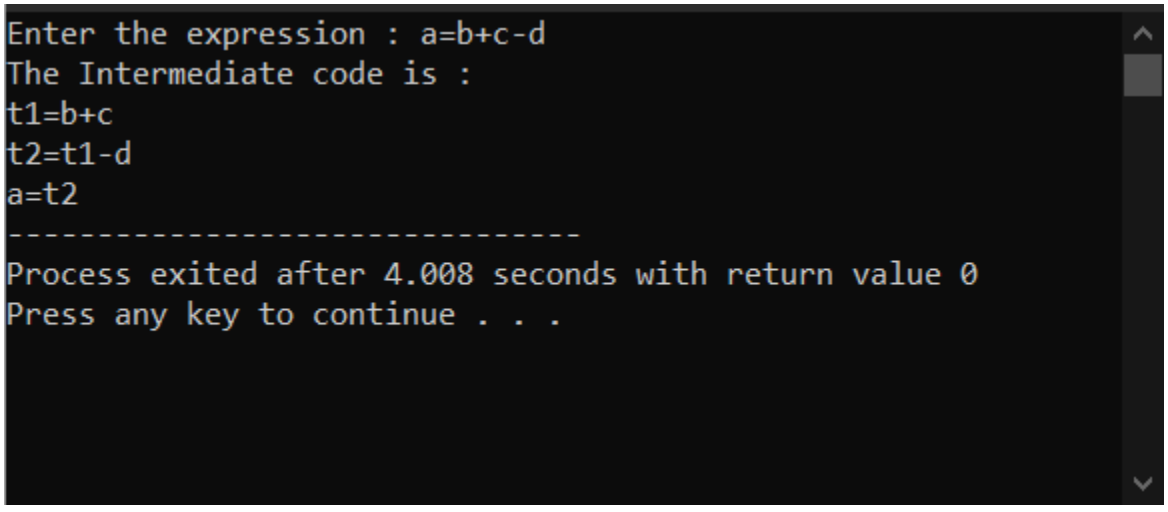
```
    }
    if(isdigit(j[i+2]) &&isdigit(j[i-2]))
    {
        a[0]='t';
        b[0]='t';
        a[1]=j[i-2];
        b[1]=j[i+2];
        sprintf(ch,"%d",c);
        j[i+2]=j[i-2]=ch[0];
    }
    if(j[i]=='*')
        printf("t%d=%s*%s\n",c,a,b);
    if(j[i]=='/')
        printf("t%d=%s/%s\n",c,a,b);
    if(j[i]=='+')
        printf("t%d=%s+%s\n",c,a,b);if(j[i]=='-')
    printf("t%d=%s-%s\n",c,a,b);
    if(j[i]=='=')
        printf("%c=t%d",j[i-1],--c);
    sprintf(ch,"%d",c);
    j[i]=ch[0];
    c++;
    small();
}
void small()
{
    pi=0;l=0;
    for(i=0;i<strlen(j);i++)
    {
        for(m=0;m<5;m++)
        if(j[i]==sw[m])
```

```
        if(pi<=p[m])
        {
           pi=p[m];
           l=1;
           k=i;
        }
    }
    if(l==1)
    dove(k);
    else
    exit(0);
}
```

Input:

a=b+c-d

Output:



Result:

A program to implement intermediate code generation - Quadruple, Triple, Indirect triple has been compiled and run successfully.