INDEX

LEXICAL ANALYZER

EX. NO. 1

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

ALGORITHM:

1. Start.
2. Get the input program from the file prog.txt.
3. Read the program line by line and check if each word in a line is a keyword, identifier, constant or an operator.
4. If the word read is an identifier, assign a number to the identifier and make an entry into the symbol table stored in sybol.txt.
5. For each lexeme read, generate a token as follows:

a. If the lexeme is an identifier, then the token generated is of the form <id, number>
b. If the lexeme is an operator, then the token generated is <op, operator>.
c. If the lexeme is a constant, then the token generated is <const, value>.
d. If the lexeme is a keyword, then the token is the keyword itself.

6. The stream of tokens generated are displayed in the console output.
7. Stop.

PROGRAM:

CODE:

week_1.cpp CD LABS

```cpp
// lexical analyser
//RA1911033010102 - Nikhil Kumar Singh
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
#include<vector>
int isKeyword(char buffer[]){
char keywords[36][10] =
{"auto","using","namespace","include","break","case","char","const","continue",
"default","do","double","else","enum","extern","float","for","goto",
"if","int","long","public","register","return","short","signed",
"sizeof","static","struct","switch","typedef","union",
"unsigned","void","volatile","while"};
int i, flag = 0;
for(i = 0; i < 32; ++i){
if(strcmp(keywords[i], buffer) == 0){
flag = 1;
break;}}
return flag;
}
char isHeader(char buffer[]){
    char headers[2][10] = {
        "iostream", "stdio"
    };

int i, flag = 0;
for(i = 0; i < 2; ++i){
if(strcmp(headers[i], buffer) == 0){
flag = 1;
break;}}
return flag;}
```

week_1.cpp

```cpp
int main(){
char ch, buffer[15], operators[] = "+-*/%=<>", special[] = "#;,.{}[]()";
FILE *fp;
int i,j=0;
fp = fopen("PROG.txt","r");
if(fp == NULL){
printf("error while opening the file\n");
exit(0);}
while((ch = fgetc(fp)) != EOF){
    for(i = 0; i < 8; ++i){
    if(ch == operators[i])
    printf("%c is operator\n", ch);
    }
    for(i = 0; i < 10; ++i){
    if(ch == special[i])
    printf("%c is special character\n", ch);
    }
    if(isdigit(ch)==true)
    printf("%c is a integer\n",ch);
    if(isalnum(ch)){
    buffer[j++] = ch;}
    else if((ch == ' ' || ch == '\n') && (j != 0)){
    buffer[j] = '\0';
    j = 0;
    if(isKeyword(buffer) == 1)
    printf("%s is keyword\n", buffer);
    else if(isHeader(buffer)==1)
    printf("%s is Header file\n", buffer);
    else
    printf("%s is identifier\n", buffer);
    }}
fclose(fp);
return 0;
}
```

OUTPUT:

```
Install the latest PowerShell for new features and improvement
s! https://aka.ms/PSWindows

PS C:\Users\Nikhil Kumar Singh\Desktop\New folder\LABS> cd "c:
\Users\Nikhil Kumar Singh\Desktop\New folder\LABS\CD LABS\" ;
if ($?) { g++ week_1.cpp -o week_1 } ; if ($?) { .\week_1 }
# is special character
include is keyword
< is operator
> is operator
iostream is Header file
using is keyword
namespace is keyword
; is special character
std is identifier
class is identifier
Student is identifier
{ is special character
public is keyword
int is keyword
; is special character
id is identifier
string is identifier
; is special character
name is identifier
} is special character
; is special character
int is keyword
main is identifier
( is special character
) is special character
{ is special character
Student is identifier
1 is a integer
; is special character
s1 is identifier
. is special character
s1id is identifier
= is operator
2 is a integer
0 is a integer
1 is a integer
; is special character
201 is identifier
1 is a integer
; is special character
Jaiswal is identifier
cout is identifier
< is operator
< is operator
1 is a integer
. is special character
s1id is identifier
< is operator
< is operator
; is special character
endl is identifier
cout is identifier
< is operator
< is operator
1 is a integer
. is special character
s1name is identifier
< is operator
< is operator
; is special character
endl is identifier
return is keyword
0 is a integer
0 is identifier
; is special character
} is special character
```

RESULT :

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

EX. NO. 2 **RE to NFA**

AIM: To convert 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

PROGRAM:

CODE:

```c
//re to nfa
//RA1911033010102 - Nikhil Kumar Singh

#include<stdio.h>
#include<string.h>
int main()
{printf("enter the RE: ");
    char reg[20];
    int q[20][3],i,j,len,a,b;
    for(a=0;a<20;a++)
    {
        for(b=0;b<3;b++)
        {
            q[a][b]=0;
        }
    }
    scanf("%s",reg);
    len=strlen(reg);
    i=0;
j=1;
    while(i<len)
    {
        if(reg[i]=='a'&&reg[i+1]!='|'&&reg[i+1]!='*')
        {
            q[j][0]=j+1;
            j++;
        }
        if(reg[i]=='b'&&reg[i+1]!='|'&&reg[i+1]!='*')
        {
            q[j][1]=j+1;
            j++;
        }
        if(reg[i]=='e'&&reg[i+1]!='|'&&reg[i+1]!='*')
        {
            q[j][2]=j+1;
            j++;
        }
        if(reg[i]=='a'&&reg[i+1]=='|'&&reg[i+2]=='b')
        {
            q[j][2]=((j+1)*10)+(j+3);
            j++;
            q[j][0]=j+1;
            j++;
            q[j][2]=j+3;
            j++;
            q[j][1]=j+1;
            j++;
            q[j][2]=j+1;
            j++;
            i=i+2;
        }
        if(reg[i]=='b'&&reg[i+1]=='|'&&reg[i+2]=='a')
        {
            q[j][2]=((j+1)*10)+(j+3);
            j++;
            q[j][1]=j+1;
            j++;
            q[j][2]=j+3;
            j++;
            q[j][0]=j+1;
            j++;
            q[j][2]=j+1;
            j++;
            i=i+2;
        }
        if(reg[i]=='a'&&reg[i+1]=='*')
```

week_2.c - LABS - Visual Studio Code

```c
        if(reg[i]=='a'&&reg[i+1]=='*')
        {
            q[j][2]=((j+1)*10)+(j+3);
            j++;
            q[j][0]=j+1;
            j++;
            q[j][2]=((j+1)*10)+(j-1);
            j++;
        }
        if(reg[i]=='b'&&reg[i+1]=='*')
        {
            q[j][2]=((j+1)*10)+(j+3);
            j++;
            q[j][1]=j+1;
            j++;
            q[j][2]=((j+1)*10)+(j-1);
            j++;
        }
        if(reg[i]==')'&&reg[i+1]=='*')
        {
            q[0][2]=((j+1)*10)+1;
            q[j][2]=((j+1)*10)+1;
            j++;
        }
        i++;
    }
    printf("Transition function \n");
    for(i=0;i<=j;i++)
    {
        if(q[i][0]!=0)
            printf("\n q[%d,a]--->%d",i,q[i][0]);
        if(q[i][1]!=0)
            printf("\n q[%d,b]--->%d",i,q[i][1]);
        if(q[i][2]!=0)
        {
            if(q[i][2]<10)
                printf("\n q[%d,e]--->%d",i,q[i][2]);
            else
                printf("\n q[%d,e]--->%d & %d",i,q[i][2]/10,q[i][2]%10);
        }
    }
    return 0;
}
```

OUTPUT :

week_2.c - LABS - Visual Studio Code

DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

```
ek_2 } ; if ($?) { .\week_2 }
enter the RE: a*b
Transition function

 q[1,e]--->2 & 4
 q[2,a]--->3
 q[3,e]--->4 & 2
 q[4,b]--->5
PS C:\Users\Nikhil Kumar Singh\Desktop\New folder\LABS\CD LABS>
```

RESULT :

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

EX. NO. 3

AIM: To convert NFA to DFA

ALGORITHM:

1. Start

2. Get the input from the user

3. Set the only state in SDFA to “unmarked”.

4. while SDFA contains an unmarked state do:

a. Let T be that unmarked state

b. for each a in % do S = e-Closure(MoveNFA(T,a))

c. if S is not in SDFA already then, add S to SDFA (as an “unmarked” state)

d. Set MoveDFA(T,a) to S

5. 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

6. Print the result.

7. Stop the program

PROGRAM:

CODE:

File Edit Selection View Go Run Terminal Help week_3.py - LABS - Visual Studio Code

week_3.py

```python
#nfa to dfa
#RA1911033010102 - Nikhil Kumar Singh
import pandas as pd
nfa = {}
n = int(input("No. of states : "))
t = int(input("No. of transitions : "))
for i in range(n):
    state = input("state name : ")
    nfa[state] = {}
    for j in range(t):
        path = input("path : ")
        print("Enter end state from state {} travelling through path {} : ".format
        reaching_state = [x for x in input().split()]
        nfa[state][path] = reaching_state

print("\nNFA :- \n")
print(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()]
new_states_list = []
dfa = {}
keys_list = list(
    list(nfa.keys())[0])
path_list = list(nfa[keys_list[0]].keys())
dfa[keys_list[0]] = {}
for y in range(t):
    var = "".join(nfa[keys_list[0]][
                path_list[y]])
    dfa[keys_list[0]][path_list[y]] = var
```

week_3.py

```python
        new_states_list.append(var)
        keys_list.append(var)
while len(new_states_list) != 0:
    dfa[new_states_list[0]] = {}
    for _ in range(len(new_states_list[0])):
        for i in range(len(path_list)):
            temp = []
            for j in range(len(new_states_list[0])):
                temp += nfa[new_states_list[0][j]][path_list[i]]
            s = ""
            s = s.join(temp)
            if s not in keys_list:
                new_states_list.append(s)
                keys_list.append(s)
            dfa[new_states_list[0]][path_list[i]] = s

    new_states_list.remove(new_states_list[0])
print("\nDFA :- \n")
print(dfa)
print("\n 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)
```

3.9.7 64-bit

OUTPUT:

File Edit Selection View Go Run Terminal Help week_3.py - LABS - Visual Studio Code

DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

```
No. of states : 3
No. of transitions : 2
state name : A
path : 0
Enter end state from state A travelling through path 0 :
A
path : 1
Enter end state from state A travelling through path 1 :
AB
state name : B
path : 0
Enter end state from state B travelling through path 0 :
C
path : 1
Enter end state from state B travelling through path 1 :
C
state name : C
path : 0
Enter end state from state C travelling through path 0 :

path : 1
Enter end state from state C travelling through path 1 :


{'A': {'0': ['A'], '1': ['AB']}, 'B': {'0': ['C'], '1': ['C']}, 'C': {'0': [], '1': []}}

Printing NFA table :-
     0     1
A  [A]  [AB]
B  [C]   [C]
C   []    []
Enter final state of NFA :
C

DFA :-

{'A': {'0': 'A', '1': 'AB'}, 'AB': {'0': 'AC', '1': 'ABC'}, 'AC': {'0': 'A', '1': 'AB'}, 'ABC': {'0': 'AC', '1': 'ABC'}}

 DFA table :-
      0    1
A     A   AB
AB   AC  ABC
AC    A   AB
ABC  AC  ABC

Final states of the DFA are :  ['AC', 'ABC']
```

RESULT :

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

ELIMINATION OF LEFT RECURSION

EX. NO. 4(a)

AIM: A program for Elimination of Left Recursion.

ALGORITHM:

1. Start the program.

2. Initialize the arrays for taking input from the user.

3. Prompt the user to input the no. of non-terminals having left recursion and no. of productions for these non-terminals.

4. Prompt the user to input the production for non-terminals.

5. 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

6. After eliminating the left recursion by applying these rules, display the productions without left recursion.

7. Stop.

CODE :

File Edit Selection View Go Run Terminal Help week_4_Lr.cpp - LABS - Visual Studio Code

week_4_Lr.cpp

```cpp
#include <iostream>
#include <vector>
#include <string>
using namespace std;
int main()
{int n;
 cout<<"\n number of non terminals ";
 cin>>n;
 cout<<"\n Enter non terminals : ";
 int i;
 vector<string> nonter(n);
 vector<int> leftrecr(n,0);
 for(i=0;i<n;++i) {
 cout<<"\n Non 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<<"\nOn 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.subst
 leftrecr[i]=1;}
 prod.push_back(temp);}
 for(i=0;i<n;++i) {
```

week_4_Lr.cpp

```cpp
 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(pr
()))==0)
  {string abc=prod[i][j].substr(nonter[i].length(),prod[i][j].le
 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 non-terminals: ";
 for(i=0;i<nonter.size();++i)
 cout<<nonter[i]<<" ";
 cout<<"\n\nNew 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;}
```

EXPLORER

LABS
- .vscode
- AI LABS
- CD LABS
  - PROG.txt
  - programm.txt
  - tempCodeRunnerFil...
  - week_1.cpp
  - week_1.exe
  - week_2.c
  - week_2.exe
  - week_3.py
  - week_4_LF.cpp
  - week_4_LF.exe
  - week_4_Lr.cpp
  - week_4_Lr.exe
  - week_5_first.cpp
  - week_5_first.exe
  - week_5_predic.cpp
  - week_5_predic.exe
  - week_6_shift.exe
  - week_6_shift.py
  - week_lead.cpp
  - week_lead.exe
  - week_lr0.cpp
  - week_lr0.exe
  - week_postfix.py
  - week_quad.c

OUTLINE

OUTPUT:

File Edit Selection View Go Run Terminal Help week_4_Lr.cpp - LABS - Visual Studio C

DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

```
 number of non terminals 3

 Enter non terminals :
 Non terminal 1 : E

 Non terminal 2 : T

 Non terminal 3 : F

Enter '^' for null
Number of E productions: 2

On enter all E productions
RHS of production 1: E+T

RHS of production 2: T

Number of T productions: 2

On enter all T productions
RHS of production 1: T*F

RHS of production 2: F

Number of F productions: 2

On enter all F productions
RHS of production 1: (E)

RHS of production 2: i
110


New non-terminals: E T F E' T'

New productions:
E -> TE'
T -> FT'
F -> (E)
F -> i
E' -> +TE'
E' -> ^
T' -> *FT'
T' -> ^
PS C:\Users\Nikhil Kumar Singh\Desktop\New folder\LABS\CD LABS>
```

RESULT : A program for implementation Of Left Recursion was compiled and run successfully.

ELIMINATION OF LEFT RECURSION

EX. NO. 4(a)

AIM: A program for Elimination of Left Recursion.

ALGORITHM:

1. Start the program.

2. Initialize the arrays for taking input from the user.

3. Prompt the user to input the no. of non-terminals having left recursion and no. of productions for these non-terminals.

4. Prompt the user to input the production for non-terminals.

5. 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

6. After eliminating the left recursion by applying these rules, display the productions without left recursion.

7. Stop.

CODE :

File Edit Selection View Go Run Terminal Help week_4_Lr.cpp - LABS - Visual Studio Code

week_4_Lr.cpp

```cpp
#include <iostream>
#include <vector>
#include <string>
using namespace std;
int main()
{int n;
 cout<<"\n number of non terminals ";
 cin>>n;
 cout<<"\n Enter non terminals : ";
 int i;
 vector<string> nonter(n);
 vector<int> leftrecr(n,0);
 for(i=0;i<n;++i) {
 cout<<"\n Non 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<<"\nOn 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.subst
 leftrecr[i]=1;}
 prod.push_back(temp);}
 for(i=0;i<n;++i) {
```

week_4_Lr.cpp

```cpp
 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(pr
()))==0)
  {string abc=prod[i][j].substr(nonter[i].length(),prod[i][j].le
 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 non-terminals: ";
 for(i=0;i<nonter.size();++i)
 cout<<nonter[i]<<" ";
 cout<<"\n\nNew 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;}
```

EXPLORER

LABS
- .vscode
- AI LABS
- CD LABS
  - PROG.txt
  - programm.txt
  - tempCodeRunnerFil...
  - week_1.cpp
  - week_1.exe
  - week_2.c
  - week_2.exe
  - week_3.py
  - week_4_LF.cpp
  - week_4_LF.exe
  - week_4_Lr.cpp
  - week_4_Lr.exe
  - week_5_first.cpp
  - week_5_first.exe
  - week_5_predic.cpp
  - week_5_predic.exe
  - week_6_shift.exe
  - week_6_shift.py
  - week_lead.cpp
  - week_lead.exe
  - week_lr0.cpp
  - week_lr0.exe
  - week_postfix.py
  - week_quad.c

OUTLINE

OUTPUT:

File Edit Selection View Go Run Terminal Help week_4_Lr.cpp - LABS - Visual Studio C

DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

```
 number of non terminals 3

 Enter non terminals :
 Non terminal 1 : E

 Non terminal 2 : T

 Non terminal 3 : F

Enter '^' for null
Number of E productions: 2

On enter all E productions
RHS of production 1: E+T

RHS of production 2: T

Number of T productions: 2

On enter all T productions
RHS of production 1: T*F

RHS of production 2: F

Number of F productions: 2

On enter all F productions
RHS of production 1: (E)

RHS of production 2: i
110


New non-terminals: E T F E' T'

New productions:
E -> TE'
T -> FT'
F -> (E)
F -> i
E' -> +TE'
E' -> ^
T' -> *FT'
T' -> ^
PS C:\Users\Nikhil Kumar Singh\Desktop\New folder\LABS\CD LABS>
```

RESULT : A program for implementation Of Left Recursion was compiled and run successfully.

FIRST AND FOLLOW

AIM: To write a program to perform first and follow using any language.

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).

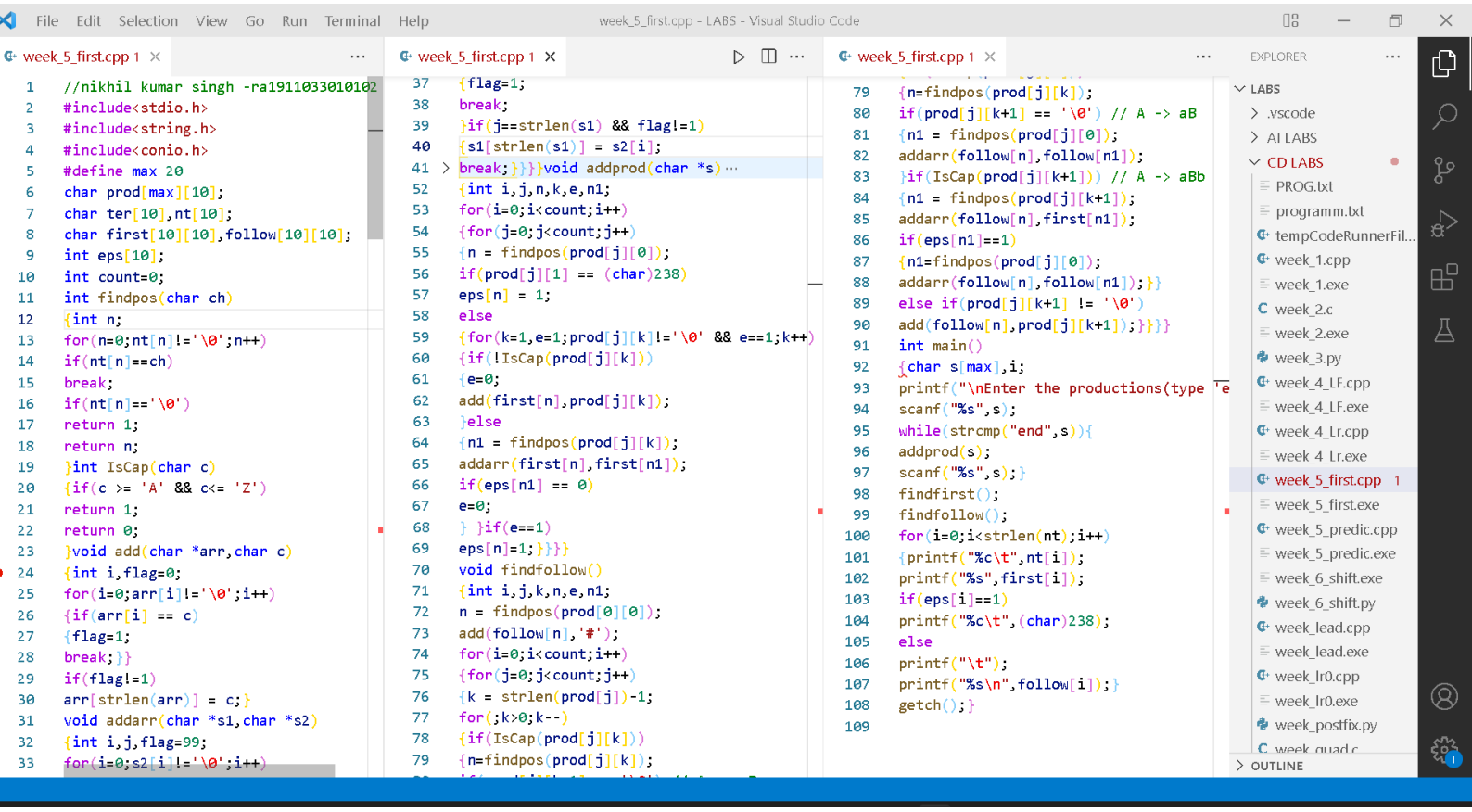
For computing the follow:

1. Always check the right side of the productions for a non-terminal, whose FOLLOW set is being found. (never see the left side).

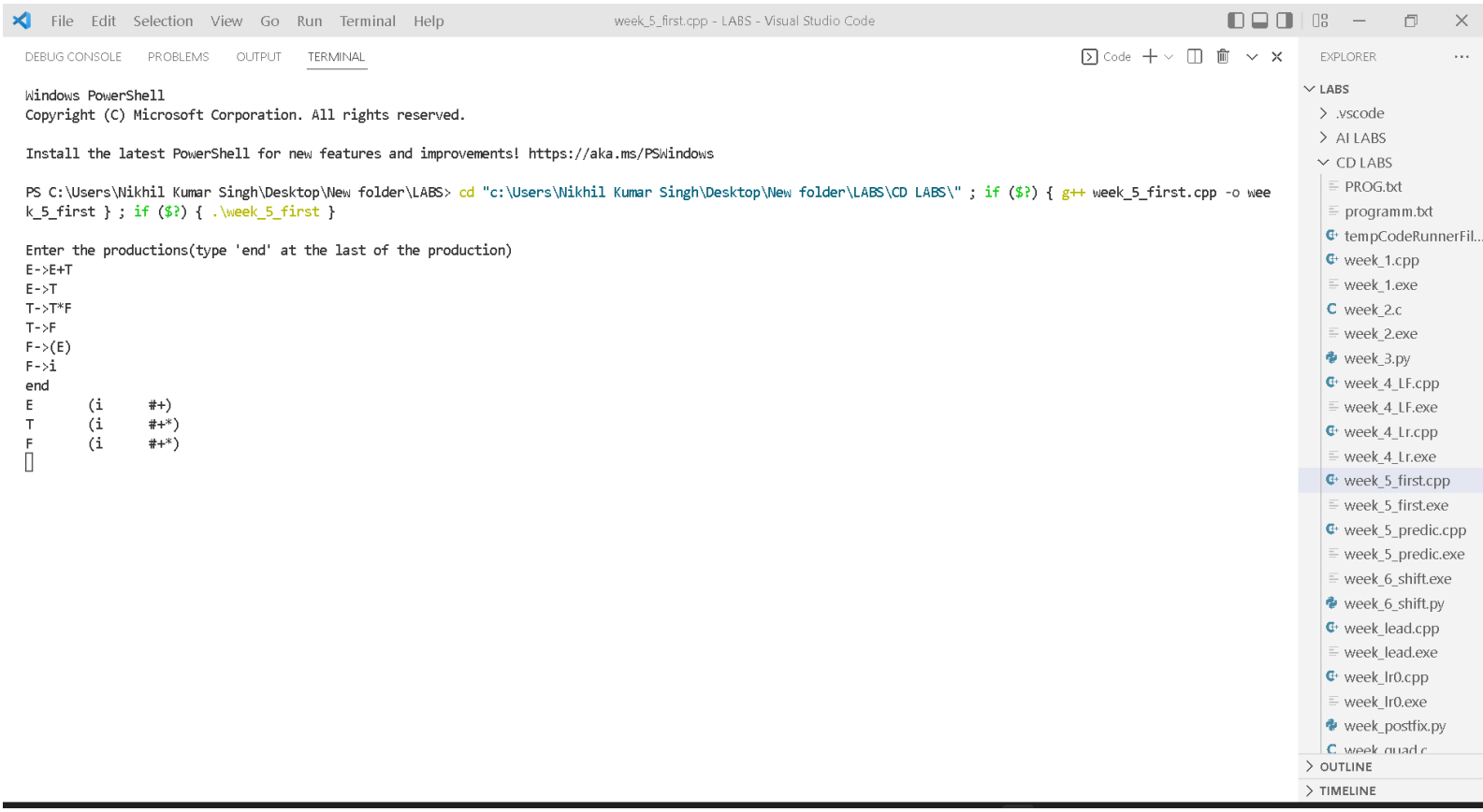
2. (a) If that non-terminal (S,A,B…) is followed by any terminal (a,b…,*,+,(,)…) , then add that terminal into the FOLLOW set.

(b) If that non-terminal is followed by any other non-terminal then add FIRST of other nonterminal into the FOLLOW set.

CODE :



OUTPUT:



RESULT :

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

PREDICTIVE PARSING

EXPERIMENT-6

Aim: A program for 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 :

File Edit Selection View Go Run Terminal Help week_5_predic.cpp - LABS - Visual Studio Code

week_5_first.cpp | week_5_predic.cpp

```cpp
//nikhil kumar singh -ra1911033010102
#include<stdio.h>
#include<conio.h>
#include<string.h>
int 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 productions\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;
}
```

week_5_predic.cpp

```cpp
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 pos\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++)
```

week_5_predic.cpp

```cpp
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:p++;
}
else
{
while(st[k][j]!=st[a][0])
{
a++;
}
p=0;
```

week_5_predic.cpp - LABS - Visual Studio Code

week_5_first.cpp | week_5_predic.cpp

```cpp
{
a++;
}
p=0;
while(ft[a][p]!='\0')
{
if(ft[a][p]!='e')
{
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])
{
while(st[k][0]!=st[a][0])
{
a++;
}
while((fol[a][c]!='\0')&&(st[a][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 pos\n");
for(i=0;i<n;i++)
printf("FOLLOW[%c]=%s\n",st[i][0]
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++)
for(p=0;p<=2;p++)
{
fin[s][p]=st[i][p];
}
t=j;
for(p=3;((st[i][j]!='|')&&(st[i][
{
fin[s][p]=st[i][j];
j++;
}
fin[s][p]='\0';
if(st[i][t]=='e')
{
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],f
b++;
}
}
else if(!((st[i][t]>64)&&(st[i][t
printf("M[%c,%c]=%s\n",st[i][0],s
else
{
b=0;
a=0;
while(st[a][0]!=st[i][3])
{
while(st[a][0]!=st[i][3])
{
a++;
}
while(ft[a][b]!='\0')
{
printf("M[%c,%c]=%s\n",st[i][0],f
b++;
}
}
s++;
}
if(st[i][j]=='|')
j++;
}
}
return 0;
}
```

OUTPUT:

week_5_predic.cpp - LABS - Visual Studio Code

DEBUG CONSOLE | PROBLEMS | OUTPUT | TERMINAL

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Nikhil Kumar Singh\Desktop\New folder\LABS> cd "c:\Users\Nikhil Kumar Singh\Desktop\New folder\LABS\CD LABS\" ; if (
$?) { g++ week_5_predic.cpp -o week_5_predic } ; if ($?) { .\week_5_predic }
enter the no. of productions
2
enter the productions in a grammar
S->CC
C->cC|d
first pos
FIRS[S]=cd
FIRS[C]=cd
follow pos
FOLLOW[S]=$
FOLLOW[C]=cd$

M[S,c]=S->CC
M[S,d]=S->CC
M[C,c]=C->cC
M[C,d]=C->d
PS C:\Users\Nikhil Kumar Singh\Desktop\New folder\LABS\CD LABS>
```

Python
Code

EXPLORER
LABS
.vscode
AI LABS
CD LABS
PROG.txt
programm.txt
tempCodeRunnerFil...
week_1.cpp
week_1.exe
week_2.c
week_2.exe
week_3.py
week_4_LF.cpp
week_4_LF.exe
week_4_Lr.cpp
week_4_Lr.exe
week_5_first.cpp
week_5_first.exe
week_5_predic.cpp
week_5_predic.exe
week_6_shift.exe
week_6_shift.py
week_lead.cpp
week_lead.exe
week_lr0.cpp
week_lr0.exe
week_postfix.py
OUTLINE
TIMELINE

Result:-

The program was successfully compiled and run

Shift Reduce

Aim: A program to implement Shift Reduce

Shift Reduce parser attempts for the construction of parse in a similar manner as done in bottom up parsing i.e. the parse tree is constructed from leaves(bottom) to the root(up). A more general form of shift reduce parser is LR parser.
This parser requires some data structures i.e.

- A input buffer for storing the input string.
- A stack for storing and accessing the production rules.

Basic Operations –

- Shift: This involves moving of symbols from input buffer onto the stack.
- Reduce: If the handle appears on top of the stack then, its reduction by using appropriate production rule is done i.e. RHS of production rule is popped out of stack and LHS of production rule is pushed onto the stack.
- Accept: If only start symbol is present in the stack and the input buffer is empty then, the parsing action is called accept. When accept action is obtained, it is means successful parsing is done.
- Error: This is the situation in which the parser can neither perform shift action nor reduce action and not even accept action.

CODE :

File Edit Selection View Go Run Terminal Help week_6_shift.py - LABS - Visual Studio Code

week_6_shift.py

```python
#Shift reduce parsing
from collections import defaultdict

n = int(input('Enter the number of production rules: '))
(variable) memo: defaultdict[Any, list]
memo = defaultdict(list)

startsymbol = ''
print('Enter the production rules:')
for i in range(n):
    inp = input()
    left, _,right = inp.partition('->')
    left = left.strip()
    right = right.strip()
    lexemes = right.split('|')
    if i == 0:
        start_symbol = left
    for c in lexemes:
        memo[left].append(c)


inp = input('Enter the input string: ')
inp = inp + '$'
i = 0
s = '$'
print(f'{"Stack Contents": <15}'+" | "+f'{"Input Buffer": <15}'+ " | "+f'Par
accepted = True
action = ''
while i<len(inp)-1 or len(s) != 2:
    action = ''
    found = False
    for key, values in memo.items():
        for value in values:
            if len(value) <= len(s) and value == s[(len(s)-len(value)):]:
```

week_6_shift.py

```python
accepted = True
action = ''
while i<len(inp)-1 or len(s) != 2:
    action = ''
    found = False
    for key, values in memo.items():
        for value in values:
            if len(value) <= len(s) and value == s[(len(s)-len(value)):]:
                action = 'Reduce ' + key + ' -> ' + value
                print(f'{s: <15}'+" | "+f'{inp[i:]: <15}'+ " | "+f'{action}')
                s = s[:-len(value)]
                s = s + key
                found = True

    if not found and i<len(inp):
        action = 'Shift'
        print(f'{s: <15}'+" | "+f'{inp[i:]: <15}'+ " | "+f'{action}')
        s = s + inp[i]
        i += 1

    if action == '':
        accepted = False
        break

if accepted:
    print(f'{s: <15}'+" | "+f'{inp[i:1]: <15}'+ " | "+f'Accepted')
else:
    print(f'{s: <15}'+" | "+f'{inp[i:]: <15}'+ " | "+f'Rejected')
```

OUTPUT:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Nikhil Kumar Singh\Desktop\New folder\LABS> & "C:/Users/Nikhil Kumar Singh/AppData/Local/Programs/Python/Python39/python.exe" "c:/Users/Nikhil Kumar Singh/Desktop/New folder/LABS/CD LABS/week_6_shift.py"
Enter the number of production rules: 3
Enter the production rules:
E->E+E
E->E*E
E->i
Enter the input string: i+i*i
Stack Contents  | Input Buffer    | Parsing Action
$               | i+i*i$          | Shift
$i              | +i*i$           | Reduce E -> i
$E              | +i*i$           | Shift
$E+             | i*i$            | Shift
$E+i            | *i$             | Reduce E -> i
$E+E            | *i$             | Reduce E -> E+E
$E              | *i$             | Shift
$E*             | i$              | Shift
$E*i            | $               | Reduce E -> i
$E*E            | $               | Reduce E -> E*E
$E              |                 | Accepted
PS C:\Users\Nikhil Kumar Singh\Desktop\New folder\LABS>
```

RESULT :

The program was successfully compiled and run.

LEADING AND TRAILING

LAB EXP 8

AIM : A program to implement 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 or End 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

CODE :

File Edit Selection View Go Run Terminal Help week_lead.cpp - LABS - Visual Studio Code

week_lead.cpp

```cpp
// nikhil kumar singh - RA1911033010102
#include<iostream>
#include<conio.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
using namespace std;

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<<"\nLEADING 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<<"\nPRODUCTION DETAILS\n";
```

week_lead.cpp

```cpp
        {
            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])
                    {
```

EXPLORER

LABS

CD LABS

PROG.txt
programm.txt
tempCodeRunnerFil...
week_1.cpp
week_1.exe
week_2.c
week_2.exe
week_3.py
week_4_LF.cpp
week_4_LF.exe
week_4_Lr.cpp
week_4_Lr.exe
week_5_first.cpp
week_5_first.exe
week_5_predic.cpp
week_5_predic.exe
week_6_shift.exe
week_6_shift.py
week_lead.cpp
week_lead.exe
week_lr0.cpp
week_lr0.exe
week_postfix.py
week_quad.c
week_quad.exe

OUTLINE

TIMELINE

week_lead.cpp - LABS - Visual Studio Code

week_lead.cpp

```cpp
                    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++)
```

week_lead.cpp

```cpp
            {
                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<vars;i++)
    {
        cout<<"\nTRAILING("<<gram[i].lhs<<") = ";
        for(j=0;j<terms;j++)
        {
            if(trail[i][j]==1)
                cout<<term[j]<<",";     }   }}
int main()
{   get();
    leading();
    trailing();
    display();}
```

EXPLORER

LABS

CD LABS

PROG.txt
programm.txt
tempCodeRunnerFil...
week_1.cpp
week_1.exe
week_2.c
week_2.exe
week_3.py
week_4_LF.cpp
week_4_LF.exe
week_4_Lr.cpp
week_4_Lr.exe
week_5_first.cpp
week_5_first.exe
week_5_predic.cpp
week_5_predic.exe
week_6_shift.exe
week_6_shift.py
week_lead.cpp
week_lead.exe
week_lr0.cpp
week_lr0.exe
week_postfix.py
week_quad.c
week_quad.exe

OUTLINE

TIMELINE

OUTPUT:

week_lead.cpp - LABS - Visual Studio Code

DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

```
PS C:\Users\Nikhil Kumar Singh\Desktop\New folder\LABS\CD LABS> cd "c:\Users\Nikhil Kumar Singh\Desktop\New folder\LABS\CD LABS\" ; if ($?) { g++ week_lead.cpp -
o week_lead } ; if ($?) { .\week_lead }

LEADING AND TRAILING

Enter the no. of variables : 3

Enter the variables :
E
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,
PS C:\Users\Nikhil Kumar Singh\Desktop\New folder\LABS\CD LABS>
```

RESULT :

The program was successfully compiled and run.

Computation of LR(0) Items

Aim: A program to implement LR(0) items

Algorithm:-1. Start.

2. Create structure for production with LHS and RHS.

3. Open file and read input from file.

4. Build state 0 from extra grammar Law S' -> S $ that is all start symbol of grammar and one Dot (.) before S symbol.

5. If Dot symbol is before a non-terminal, add grammar laws that this non-terminal is in Left Hand Side of that Law and set Dot in before of first part of Right Hand Side.

6. If state exists (a state with this Laws and same Dot position), use that instead.

7. Now find set of terminals and non-terminals in which Dot exist in before.

8. If step 7 Set is non-empty go to 9, else go to 10.

9. For each terminal/non-terminal in set step 7 create new state by using all grammar law that Dot 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.

10. Go to step 5.

11. End of state building.

12. Display the output.

13. End.

CODE :

File Edit Selection View Go Run Terminal Help week_lr0.cpp - LABS - Visual Studio Code

week_lead.cpp | week_lr0.cpp 1

```cpp
//NIKHIL KUMAR SINGH -RA1911033010102
#include<iostream>
#include<conio.h>
#include<string.h>
using namespace std;
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;B
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++)
```

week_lr0.cpp 1

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

int main()
{
cout<<"ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO END) :\n";
do
{
```

week_lr0.cpp - LABS - Visual Studio Code

week_lead.cpp | week_lr0.cpp 1

```cpp
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<<" ";

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,".");
```

week_lr0.cpp 1

```cpp
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(int 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";   }}
```

OUTPUT:

week_lr0.cpp - LABS - Visual Studio Code

DEBUG CONSOLE | PROBLEMS | OUTPUT | TERMINAL

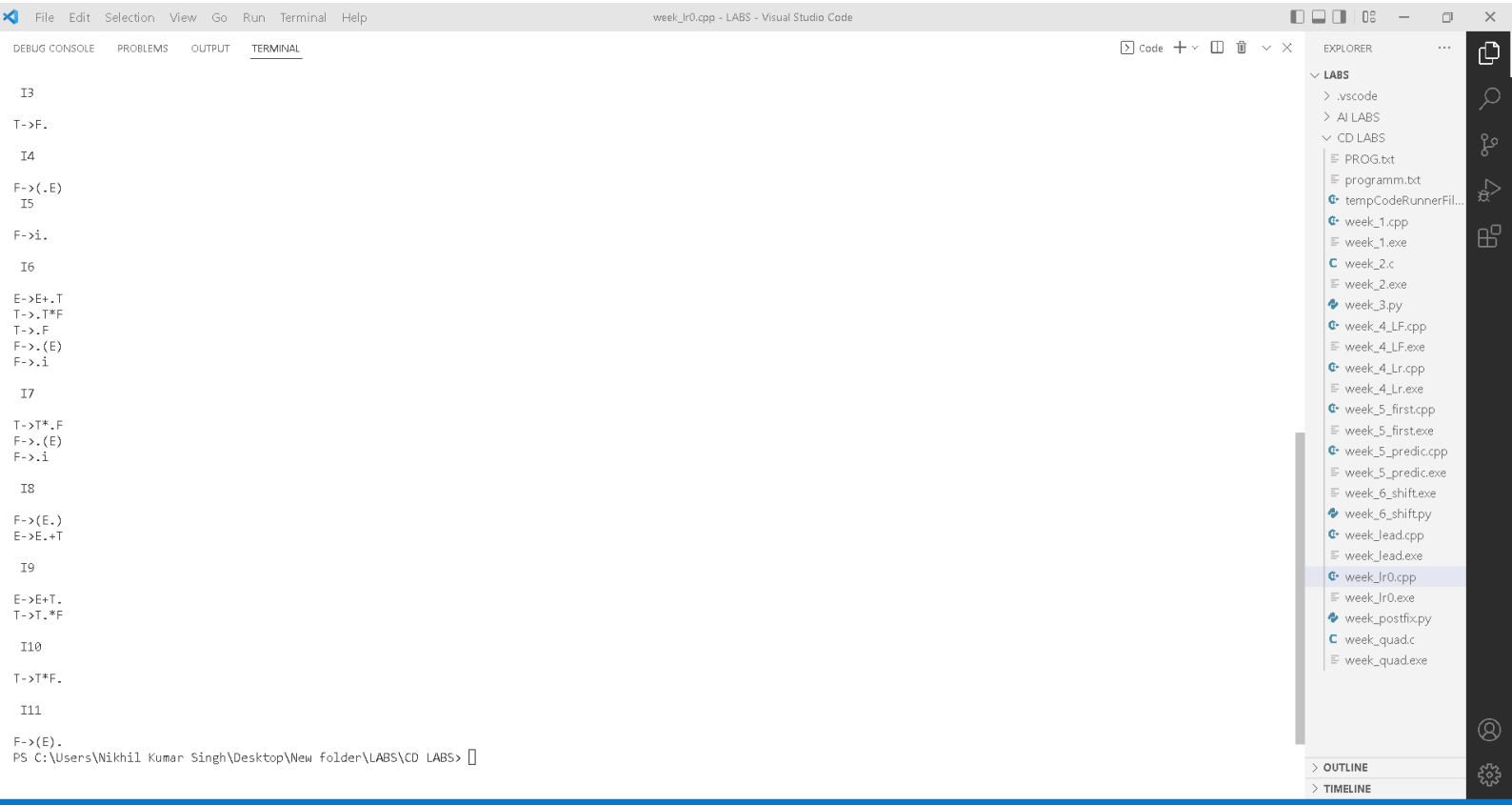
```
PS C:\Users\Nikhil Kumar Singh\Desktop\New folder\LABS\CD LABS> cd "c:\Users\Nikhil Kumar Singh\Desktop\New folder\LABS\CD LABS\" ; if ($?) { g++ week_lr0.cpp -o week_lr0 } ; if
($?) { .\week_lr0 }
ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO END) :
E->E+T
E->T
T->T*F
T->F
F->(E)
F->i
0


 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

 I1

A->E.
E->E.+T

 I2

E->T.
T->T.*F

 I3
```

EXPLORER

LABS
- .vscode
- AI LABS
- CD LABS
  - PROG.txt
  - programm.txt
  - tempCodeRunnerFil...
  - week_1.cpp
  - week_1.exe
  - week_2.c
  - week_2.exe
  - week_3.py
  - week_4_LF.cpp
  - week_4_LF.exe
  - week_4_Lr.cpp
  - week_4_Lr.exe
  - week_5_first.cpp
  - week_5_first.exe
  - week_5_predic.cpp
  - week_5_predic.exe
  - week_6_shift.exe
  - week_6_shift.py
  - week_lead.cpp
  - week_lead.exe
  - week_lr0.cpp
  - week_lr0.exe
  - week_postfix.py
  - week_quad.c
  - week_quad.exe

OUTLINE

TIMELINE



RESULT :

The program was successfully compiled and run.

Postfix, Prefix

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

Algorithm:-

1. Declare set of operators.

2. Initialize an empty stack.

3. To convert INFIX to POSTFIX follow the following steps

4. Scan the infix expression from left to right.

5. If the scanned character is an operand, output it.

6. 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.

7. 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.

8. If the scanned character is an '(', push it to the stack.

9. If the scanned character is an ')', pop the stack and output it until a '(' is encountered, and discard both the parenthesis.

10. Pop and output from the stack until it is not empty.

11. To convert INFIX to PREFIX follow the following steps

12. First, reverse the infix expression given in the problem.

13. Scan the expression from left to right.

14. Whenever the operands arrive, print them.

15. If the operator arrives and the stack is found to be empty, then simply push the operator into the stack.

16. Repeat steps 6 to 9 until the stack is empty

CODE :

week_postfix.py - LABS - Visual Studio Code

```python
OPERATORS = set(['+', '-', '*', '/', '(', ')'])
PRI = {'+': 1, '-': 1, '*': 2, '/': 2}
### INFIX ===> POSTFIX ###
#NIKHIL KUMAR SINGH - RA1911033010102
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[
                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 == '(':
```

```python
    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] <
                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:

week_postfix.py - LABS - Visual Studio Code

DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

```
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Nikhil Kumar Singh\Desktop\New folder\LABS> & "C:/Users/Nikhil Kumar Singh/AppData/Local/Programs/Python/Python39/python.exe" "c:/Users/Nikhil
 Kumar Singh/Desktop/New folder/LABS/CD LABS/week_postfix.py"
INPUT THE EXPRESSION: A+B
PREFIX: +AB
POSTFIX: AB+
PS C:\Users\Nikhil Kumar Singh\Desktop\New folder\LABS>
```

RESULT :

The program was successfully compiled and run.