Date
9/2/23

# EXPERIMENT-8

## LEADING AND TRAILING

**Aim:** To implement code for leading and Trailing

**Algorithm :**

Leading :

Y : only Non-terminal
a : terminal

① If $A \rightarrow Ya$ or $A \rightarrow Ya\beta$
lead (A) = a

② If $A \rightarrow B$
lead (A) = lead (B)

③ If $A \rightarrow aB$
Lead (A) = a

Trailing :

① If $A \rightarrow B\alpha Y$
trail (A) = x

② If $A \rightarrow B$
trail (A) = trail (B)

③ If $A \rightarrow Bx$
trail (A) = x

**PROGRAM:**

```
#include <stdio.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 ins, rns[20][20];
  };
```

```
gram [50];
void get()
{ count <<"\n leading and Trailing \n";
  count <<"\n Enter no of variable : ";
   cin >> vars;
  count <<"\n Enter the variables : \n";
for (i=0; i< vars; i++)
    {   cin >> gram [i].lhs;
        var [i] = gram [i] .ins;
    }
 Count <<"\n Enter the no of terminds : ";
   cin >> terms;
Count <<"\n Enter the terminals :";
 for (j = 0; j < terms ; j++)
        cin >> term [j];
 count <<"\n Product Details \n";
for (i=0; i < vars ;i++)
 { count << "\n Enter the no of productions of"
        <<gram [i].lhs is;
   Cin >> gram [i]. prodno;
 for (j= 0; j < gram [i] .prodno; j++)
   {  count << gram [i].lhs << "->";
      cin >> gram [i].rhs[j];
   }}
void leading ()
{ for (i=0; i< vars; i++)
   { for (j= 0; j < gram[i]. prodno ; j++)
```

```
{ lead [i][k] = 1;
  }
  else { if (gram[i].rhs[j][i] == 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] != '\0')
      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[i][count-2] == term[k]
        trail[i][k] = 1; }
    }
void display()
{ for (j = 0; j < terms; j++)
```

Output:

Leading & Trailing

Enter no of variables : 3

Enter variables :

E

T

F

Enter no of terminals : 5

Enter the terminals : +

*

i

)

(

Production details

Enter no of production of E : 2

E → E+T

E → T

Enter no of productions of T : 2

T → T*F

T → F

Enter no of productions of F : 2

F → i

F → (E)

Leading (E) = +, *, i, (

Leading (T) = *, i, (

Leading (F) = i, (

Trailing (E) = +, *, i, )

Trailing (T) = *, i, )

Trailing (F) = i, )

OP verified
19/9/23

```
    if (trail[i][j] == 1)
        count << term[j] << ","; }
}
int main() {
    get();
    leading();
    trailing();
    display(); }
```

RESULT: Hence we have successfully executed code for leading & trailing.