RA1911026010033

Compiler Design

Week - 8

Topic :

Leading And Trailing

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 :

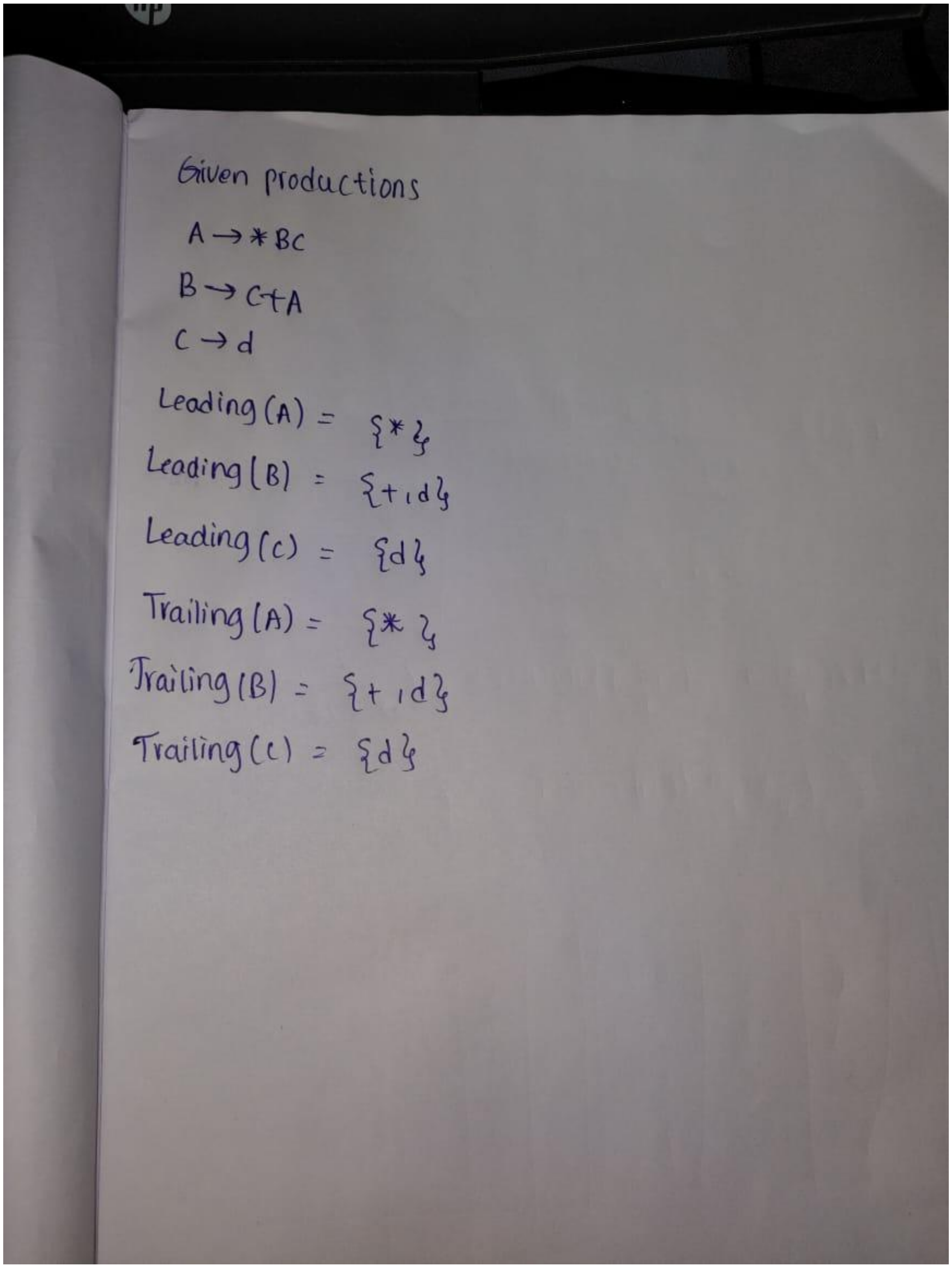
```
#include <bits/stdc++.h> using namespace std; #include <cstring>
int nt, t, top = 0;
char s[50], NT[10], T[10], st[50], l[10][10], tr[50][50];
int searchnt(char a)
{
    int count = -1, i;

    for (i = 0; i < nt; i++)
    {
        if (NT[i] == a)
            return i;
    }
    return count;
}
int searchter(char a)
{
    int count = -1, i;
    for (i = 0; i < t; i++)
    {
        if (T[i] == a)
            return i;
    }
    return count;
}
void push(char a)
{
    s[top] = a;
    top++;
```

```
}
char pop()
{
    top--;
    return s[top];
}
void installl(int a, int b)
{
    if (l[a][b] == 'f')
    {
        l[a][b] = 't';
        push(T[b]);
        push(NT[a]);
    }
}
void installt(int a, int b)
{
    if (tr[a][b] == 'f')
    {
        tr[a][b] = 't';
        push(T[b]);
        push(NT[a]);
    }
}
int main()
{
    int i, s, k, j, n;
    char pr[30][30], b, c;
    cout << "Enter the no of productions:";
    cin >> n;
    cout << "Enter the productions one by one\n";
    for (i = 0; i < n; i++)
        cin >> pr[i];
    nt = 0;
    t = 0;
    for (i = 0; i < n; i++)
    {
        if ((searchnt(pr[i][0])) == -1)
            NT[nt++] = pr[i][0];
    }
    for (i = 0; i < n; i++)
    {
        for (j = 3; j < strlen(pr[i]); j++)
        {
            if (searchnt(pr[i][j]) == -1)
            {
                if (searchter(pr[i][j]) == -1)
                    T[t++] = pr[i][j];
            }
        }
    }
    for (i = 0; i < nt; i++)
    {
        for (j = 0; j < t; j++)
            l[i][j] = 'f';
    }
    for (i = 0; i < nt; i++)
    {
        for (j = 0; j < t; j++)
            tr[i][j] = 'f';
    }
    for (i = 0; i < nt; i++)
    {
        for (j = 0; j < n; j++)
        {
            if (NT[(searchnt(pr[j][0]))] == NT[i])
            {
                if (searchter(pr[j][3]) != -1)
                    installl(searchnt(pr[j][0]), searchter(pr[j][3]));
                else
                {
                    for (k = 3; k < strlen(pr[j]); k++)
                    {
                        if (searchnt(pr[j][k]) == -1)
                        {
                            installl(searchnt(pr[j][0]), searchter(pr[j][k]));
                            break;
                        }
                    }
                }
            }
        }
    }
    while (top != 0)
    {
        b = pop();
        c = pop();
        for (s = 0; s < n; s++)
        {
            if (pr[s][3] == b)
                installl(searchnt(pr[s][0]), searchter(c));
        }
    }
    for (i = 0; i < nt; i++)
    {
        cout << "Leading[" << NT[i] << "]"
             << "\t{";
        for (j = 0; j < t; j++)
        {
            if (l[i][j] == 't')
                cout << T[j] << ",";
        }
        cout << "}\n";
    }
    top = 0;
    for (i = 0; i < nt; i++)
    {
        for (j = 0; j < n; j++)
        {
            if (NT[searchnt(pr[j][0])] == NT[i])
            {
                if (searchter(pr[j][strlen(pr[j]) - 1]) != -1)
```

```
                    installt(searchnt(pr[j][0]), searchter(pr[j][strlen(pr[j]) - 1]));
                else
                {
                    for (k = (strlen(pr[j]) - 1); k >= 3; k--)
                    {
                        if (searchnt(pr[j][k]) == -1)
                        {
                            installt(searchnt(pr[j][0]), searchter(pr[j][k]));
                            break;
                        }
                    }
                }
            }
        }
    }
    while (top != 0)
    {
        b = pop();
        c = pop();
        for (s = 0; s < n; s++)
        {
            if (pr[s][3] == b)
                installt(searchnt(pr[s][0]), searchter(c));
        }
    }
    for (i = 0; i < nt; i++)
    {
        cout << "Trailing[" << NT[i] << "]"
             << "\t{";
        for (j = 0; j < t; j++)
        {
            if (tr[i][j] == 't')
                cout << T[j] << ",";
        }
        cout << "}\n";
    }
    return 0;
}
```

Manual Calculation:-



Given productions

$A \rightarrow * Bc$

$B \rightarrow C + A$

$C \rightarrow d$

Leading(A) = {*}

Leading(B) = {+, d}

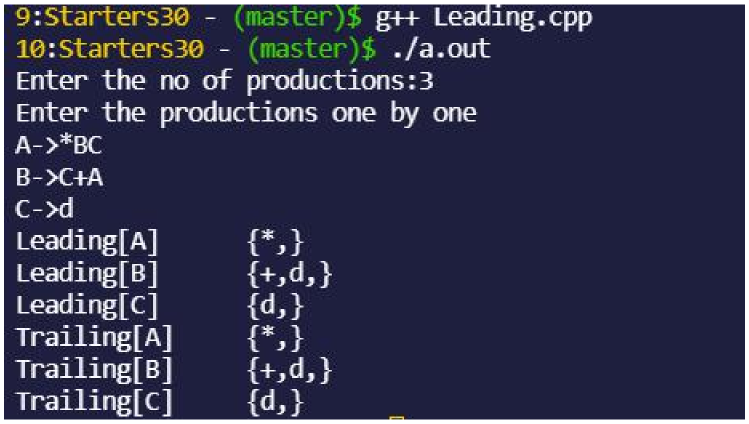
Leading(C) = {d}

Trailing(A) = {*}

Trailing(B) = {+, d}

Trailing(C) = {d}

OUTPUT



```
9:Starters30 - (master)$ g++ Leading.cpp
10:Starters30 - (master)$ ./a.out
Enter the no of productions:3
Enter the productions one by one
A->*BC
B->C+A
C->d
Leading[A]      {*,}
Leading[B]      {+,d,}
Leading[C]      {d,}
Trailing[A]     {*,}
Trailing[B]     {+,d,}
Trailing[C]     {d,}
```

RESULT :

The program was successfully compiled and run.