Ex No:5

Date:05-03-2021

First and Follow computation

AIM: To compute First and Follow for a Grammar

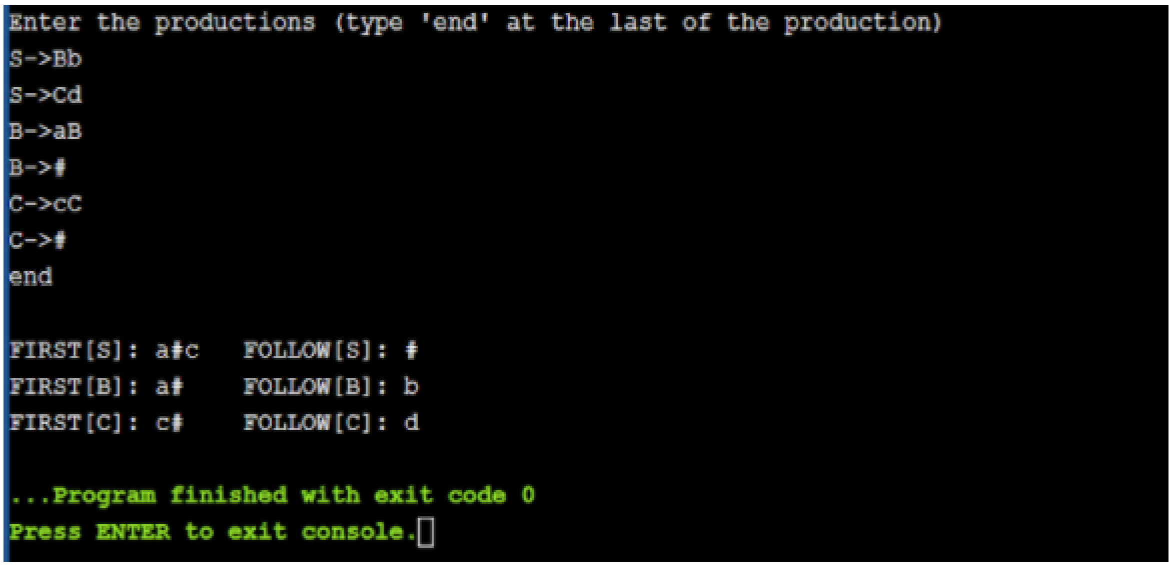
ALGORITHM:
For computing the first:
1. If X is a terminal then FIRST(X) = {X} Example: F -> I | id We can write it as FIRST(F) -> { (, id)
2. If X is a non-terminal like E -> T then to get FIRSTI substitute T with other productions until you get a terminal as the first symbol
3. If X -> ε then add ε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

Program:

```
#include
<bits/stdc++.h>
#define max 20
using namespace std;
char prod[max][10], ter[10], nt[10], first[10][10],
follow[10][10]; int eps[10],c=0;
int findpos(char ch)
{
  int n;
  for (n = 0; nt[n] != '\0';
     n++) if (nt[n] == ch)
        break;
  if (nt[n] ==
     '\0') return
     1;
  return n;
}
int IsCap(char c)
{
  return (c >= 'A' && c <= 'Z') ? 1 : 0;
}
void add(char *arr, char c)
{
  int i, flag = 0;
  for (i = 0; arr[i] != '\0';
     i++) if (arr[i] == c)
     {
        flag =
        1;
        break;
        }
    if (flag != 1)
        arr[strlen(arr)]
        = c;
}
void addarr(char *s1, char *s2)
{
    int i, j, flag = 99;
    for (i = 0; s2[i] != '\0'; i++)
    {
        flag = 0;
        for (j = 0;; j++)
        {
            if (s2[i] == s1[j])
            {
                flag =
                1;
                break;
            }
            if (j == strlen(s1) && flag != 1)
            {
                s1[strlen(s1)] =
                s2[i]; break;
            }
        }
        }
}
void addprod(char *s)
{
    int i;
    prod[c][0] = s[0];
    for (i = 3; s[i] != '\0'; i++)
    {
   if (!IsCap(s[i]))
        add(ter, s[i]);
       prod[c][i - 2] = s[i];
    }
    prod[c][i - 2] = '\0';
    add(nt,
    s[0]); c++;
}
void findfirst()
{
    int i, j, n, k, e,
    n1; for (i = 0; i <
    c; i++)
    {
        for (j = 0; j < c; j++)
        {
            n = findpos(prod[j][0]);
            if (prod[j][1] ==
                (char)238) eps[n] =
                1;
        else
        {
          for (k = 1, e = 1; prod[j][k] != '\0' && e == 1; k++)
          {
            if (!IsCap(prod[j][k]))
            {
              e = 0;
              add(first[n], prod[j][k]);
            }
            else
            {
              n1                    =
              findpos(prod[j][k]);
              addarr(first[n],
              first[n1]); if (eps[n1]
              == 0)
                e = 0;
            }
          }
          if (e == 1)
            eps[n] =
            1;
        }
      }
    }
  }
  void findfollow()
  {
    int i, j, k, n, e, n1;
    n = findpos(prod[0][0]);
    add(follow[n],
    '#'); for (i = 0; i
    < c; i++)
    {
      for (j = 0; j < c; j++)
      {
        for (k = strlen(prod[j]) - 1; k > 0; k--)
        {
        if (IsCap(prod[j][k]))
          {
            n = findpos(prod[j][k]);
            if (prod[j][k + 1] == '\0') // A -> aB
            {
              n1 = findpos(prod[j][0]);
              addarr(follow[n],
              follow[n1]);
            }
            if (IsCap(prod[j][k + 1])) // A -> aBb
            {
              n1 = findpos(prod[j][k
              + 1]); addarr(follow[n],
              first[n1]); if (eps[n1]
              == 1)
```

```
                    {
                      n1 = findpos(prod[j][0]);
                      addarr(follow[n],
                      follow[n1]);
                    }
                  }
                  else if (prod[j][k + 1] != '\0')
                    add(follow[n], prod[j][k + 1]);
                }
              }
            }
          }
        }
int main()
{
  char s[max], i;
  cout << "\nEnter the productions (type 'end' at the last of the
  production)\n"; cin >> s;
  while (strcmp("end", s))
  {
    addprod(
    s); cin >>
    s;
  }
  findfirst();
  findfollow
  ();
  for (i = 0; i < strlen(nt); i++)
  {
    cout << "\nFIRST[" << nt[i] << "]: "
    << first[i]; if (eps[i] == 1)
      cout << (char)238 <<
    "\t"; else
      cout << "\t";
    cout << "FOLLOW[" << nt[i] << "]: " << follow[i];
  }
  return 0;
}
```

Output:



```
Enter the productions (type 'end' at the last of the production)
S->Bb
S->Cd
B->aB
B->#
C->cC
C->#
end

FIRST[S]: a#c   FOLLOW[S]: #
FIRST[B]: a#    FOLLOW[B]: b
FIRST[C]: c#    FOLLOW[C]: d

...Program finished with exit code 0
Press ENTER to exit console.
```

Result: The First and Follow computation is implemented successfully.

RA1811003010303 RAGHU B