EX. NO. 6
12/03/21

PREDICTIVE PARSER

AIM: To construct a predictive parser

ALGORITHM:

- Start the program.
- Initialize the required variables.
- Get the number of coordinates and productions from the user.
- 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];
- Print the resulting stack.
- Print if the grammar is accepted or not.
- Exit the program.

PROGRAM:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
   char fin[10][20], st[10][20], ft[20][20], fol[20][20];
   int a, i, t, b, n, j, s = 0, p;
   cout << "Enter the number of productions: ";
   cin >> n;
   cout << "Enter the productions of the grammar:\n";
   for (i = 0; i < n; i++)
      cin >> st[i];
   cout << "\nEnter the FIRST and FOLLOW of each non-terminal:";
   for (i = 0; i < n; i++)
   {
      cout << "\nFIRST[" << st[i][0] << "] : ";
      cin >> ft[i];
      cout << "FOLLOW[" << st[i][0] << "] : ";
      cin >> fol[i];
   }
   cout << "\nThe contents of the predictive parser table are:\n";
   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++, j++)
                fin[s][p] = st[i][j];
            fin[s][p] = '\0';
            if (st[i][t] == 'e')
            {
                a = b = 0;
                while (st[a++][0] != st[i][0])
                    ;
                while (fol[a][b] != '\0')
                {
                    cout << "M[" << st[i][0] << "," << fol[a][b]
                        << "] = " << fin[s] << "\n";
                    b++;
                }
            }
            else if (!(st[i][t] > 64 && st[i][t] < 91))
                cout << "M[" << st[i][0] << "," << st[i][t]
                    << "] = " << fin[s] << "\n";
            else
            {
                a = b = 0;
                while (st[a++][0] != st[i][3])
                    ;
                while (ft[a][b] != '\0')
                {
                    cout << "M[" << st[i][0] << "," << ft[a][b]
                        << "] = " << fin[s] << "\n";
                    b++;
                }
            }
            s++;
        }
        if (st[i][j] == '|')
            j++;
    }
  }
  return 0;
}
```

OUTPUT:

```
Enter the number of productions: 5
Enter the productions of the grammar:
E->TD
D->+TD|e
T->FG
G->*FG|e
F->(E)|i

Enter the FIRST and FOLLOW of each non-terminal:
FIRST[E] : (i
FOLLOW[E] : $)

FIRST[D] : +e
FOLLOW[D] : $)

FIRST[T] : (i
FOLLOW[T] : +$)

FIRST[G] : *e
FOLLOW[G] : +$)

FIRST[F] : (i
FOLLOW[F] : *+$)

The contents of the predictive parser table are:
M[E,*] = E->TD
M[E,e] = E->TD
M[D,+] = D->+TD
M[D,+] = D->e
M[D,$] = D->e
M[D,)] = D->e
M[T,k] = T->FG
M[T, ] = T-
>FG
M[G,*] = G->*FG
M[G,*] = G->e
M[G,+] = G->e
M[G,$] = G->e
M[G,)] = G->e
M[F,(] = F->(E)
M[F,i] = F->i
```

RESULT: program to implement the predictive parser was compiled, executed and verified successfully.