FIRST AND FOLLOW COMPUTATION

NAME: **RISHAL RAMESH** ***EXP NO:*** **5**

REG NO: **RA1911030010084**

DATE: **24/02/2022**

AIM:

To study and implement FIRST & FOLLOW computations for the given productions.

LANGUAGE USED:

C++

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 the to get FIRST 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.

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

CODE:

```
#include<bits/stdc++.h>
using namespace std;
set<char> ss;
bool dfs(char i, char org, char last, map<char,vector<vector<char>>> &mp)
{
    bool rtake = false;
    for(auto r : mp[i])
    {
        bool take = true;
        for(auto s : r)
        {
            if(s == i)
              break;
            if(!take)
              break;
            if(!(s>='A'&&s<='Z')&&s!='e')
            {
                ss.insert(s);
                break;
            }
            else if(s == 'e')
            {
                if(org == i||i == last)
                ss.insert(s);
                rtake = true;
                break;
            }
            else
            {
```

```
                take = dfs(s,org,r[r.size()-1],mp);
                rtake |= take;
            }
        }
    }
    return rtake;
}
int main()
{
    int i,j;
    ifstream fin("inputfirstfollow.txt");
    string num;
    vector<int> fs;
    vector<vector<int>> a;
    map<char,vector<vector<char>>> mp;
    char start;
    bool flag = 0;
    cout<<"Grammar: "<<'\n';
    while(getline(fin,num))
    {
        if(flag == 0) start = num[0],flag = 1;
        cout<<num<<'\n';
        vector<char> temp;
        char s = num[0];
        for(i=3;i<num.size();i++)
        {
            if(num[i] == '|')
            {
                mp[s].push_back(temp);
                temp.clear();
```

```
        }
        else
          temp.push_back(num[i]);
    }
    mp[s].push_back(temp);
}
map<char,set<char>> fmp;
for(auto q : mp)
{
    ss.clear();
    dfs(q.first,q.first,q.first,mp);
    for(auto g : ss)
      fmp[q.first].insert(g);
}
cout<<'\n';
cout<<"FIRST: "<<'\n';
for(auto q : fmp)
{
    string ans = "";
    ans += q.first;
    ans += " = {";
    for(char r : q.second)
    {
        ans += r;
        ans += ',';
    }
    ans.pop_back();
    ans+="}";
    cout<<ans<<'\n';
}
```

```
map<char,set<char>> gmp;
gmp[start].insert('$');
int count = 10;
while(count--)
{
    for(auto q : mp)
    {
        for(auto r : q.second)
        {
            for(i=0;i<r.size()-1;i++)
            {
                if(r[i]>='A'&&r[i]<='Z')
                {
                    if(!(r[i+1]>='A'&&r[i+1]<='Z'))
                      gmp[r[i]].insert(r[i+1]);
                    else
                    {
                        char temp = r[i+1];
                        int j = i+1;
                        while(temp>='A'&&temp<='Z')
                        {
                            if(*fmp[temp].begin()=='e')
                            {
                                for(auto g : fmp[temp])
                                {
                                    if(g=='e')
                                      continue;
                                    gmp[r[i]].insert(g);
                                }
                                j++;
```

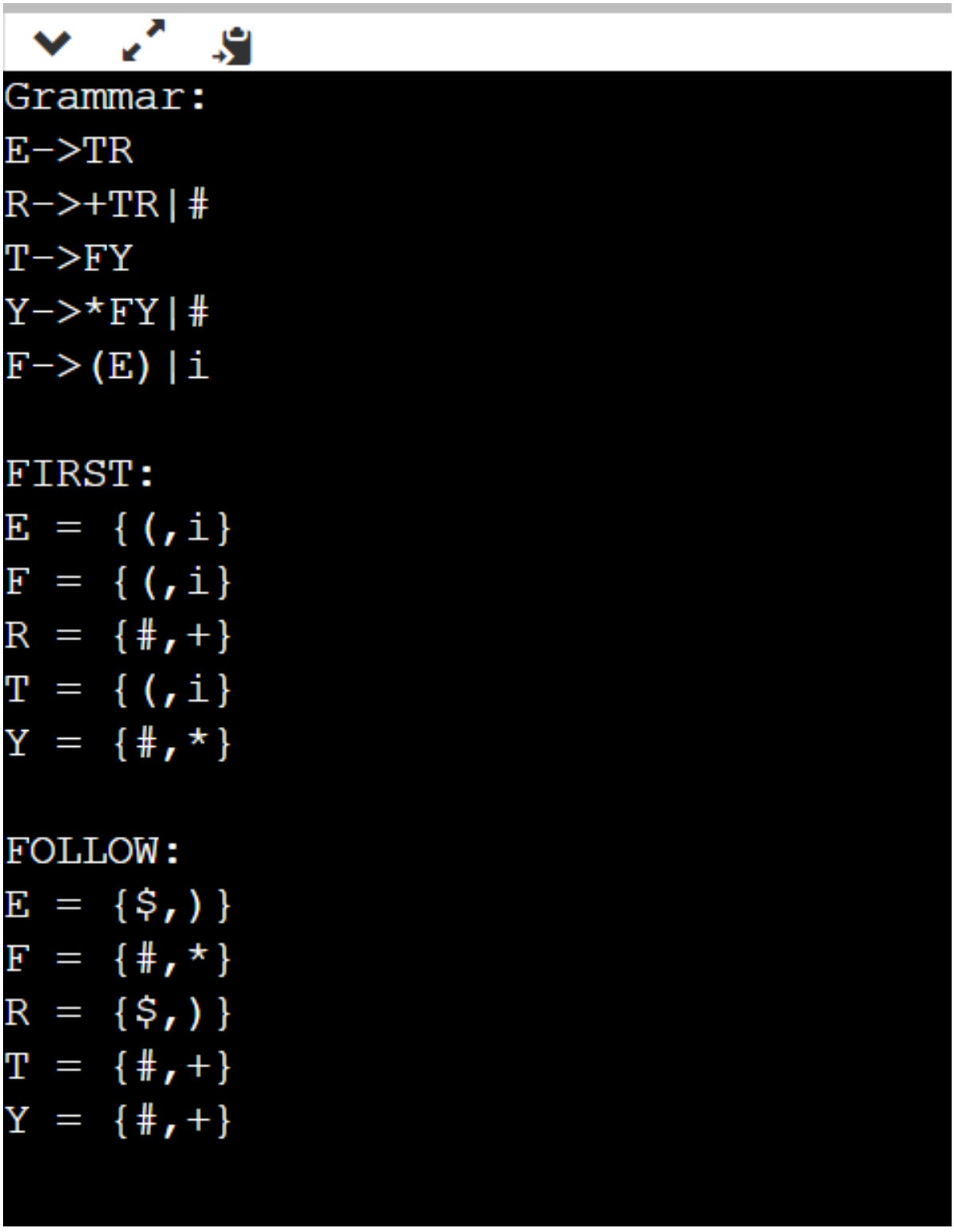
```
                    if(j<r.size())
                    {
                        temp = r[j];
                        if(!(temp>='A'&&temp<='Z'))
                        {
                            gmp[r[i]].insert(temp);
                            break;
                        }
                    }
                    else
                    {
                        for(auto g : gmp[q.first]) gmp[r[i]].insert(g);
                        break;
                    }
                }
                else
                {
                    for(auto g : fmp[temp])
                    {
                        gmp[r[i]].insert(g);
                    }
                    break;
                }
            }
        }
    }
}
if(r[r.size()-1]>='A'&&r[r.size()-1]<='Z')
{
    for(auto g : gmp[q.first]) gmp[r[i]].insert(g);
```

```
            }
        }
    }
}
cout<<'\n';
cout<<"FOLLOW: "<<'\n';
for(auto q : gmp)
{
    string ans = "";
    ans += q.first;
    ans += " = {";
    for(char r : q.second)
    {
        ans += r;
        ans += ',';
    }
    ans.pop_back();
    ans+="}";
    cout<<ans<<'\n';
}
return 0;
}
```

inputfirstfollow.txt:

```
E->TR
R->+TR|#
T->FY
Y->*FY|#
F->(E)|i
```

OUTPUT:



RESULT:

The FIRST & FOLLOW computation was studied and executed successfully in C++.