Compiler Design

Week - 5

Topic :

First And Follow Computation

Aim:

To Implement Program to calculate First and Follow of grammar.

Algorithm:

First:

To compute FIRST(X), where X is a grammar symbol,
● If X is a terminal, then FIRST(X) = {X}.
● If X → ε is a production, then add ε to FIRST(X).
● If X is a non-terminal and X → Y1 Y2 ⋯ Yk is a production, then add
FIRST(Y1) to FIRST(X). If Y1 derives ε, then add FIRST(Y2) to FIRST(X).

Follow :

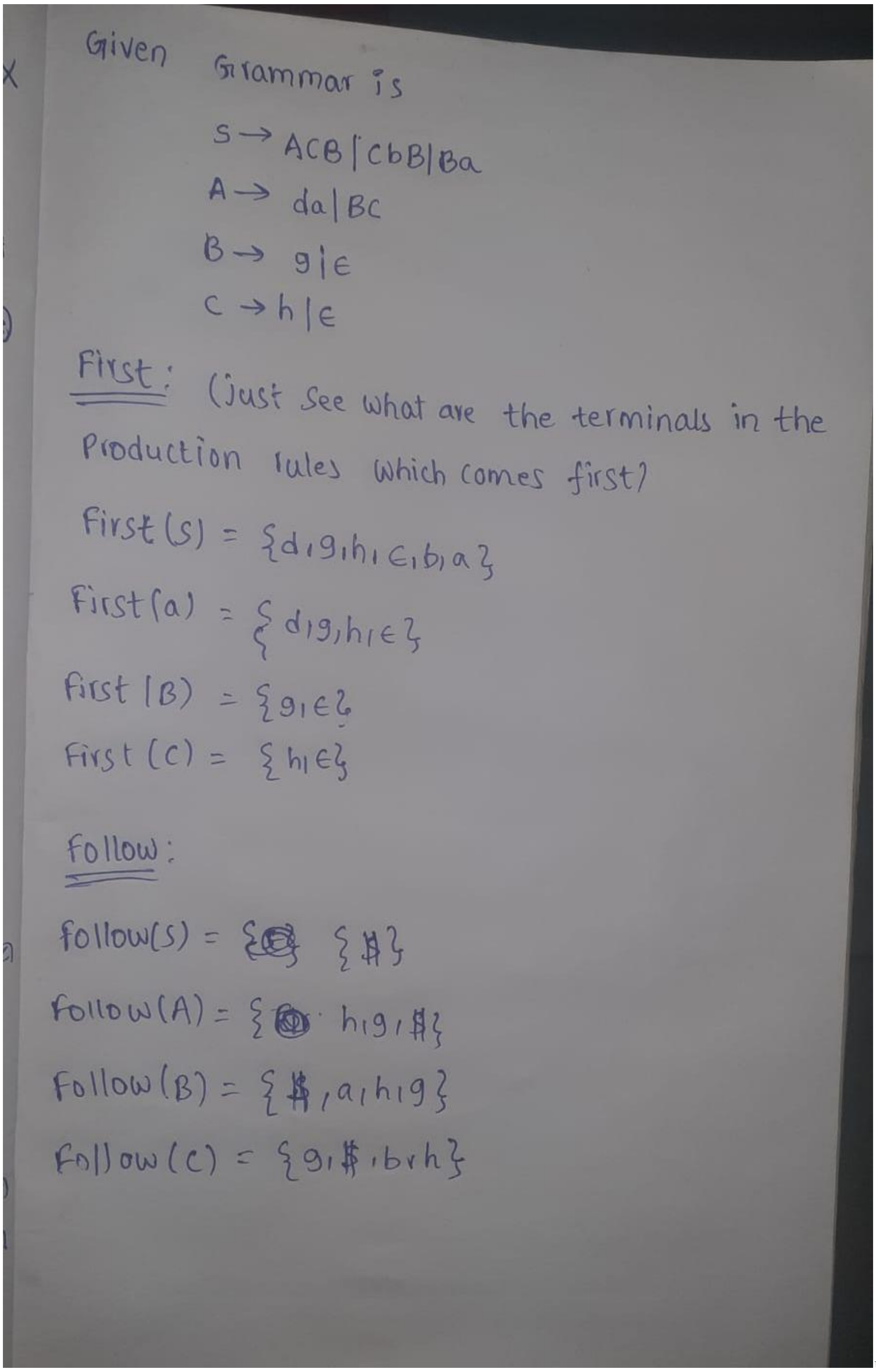
To compute FOLLOW(X), where X is a grammar symbol,
● For the FOLLOW(start symbol) place $, where $ is the input end marker.
● If there is a production A → αBβ, then everything in FIRST(β) except ε is
in FOLLOW(B).
● If there is a production A → αB, or a production A → αBβ where FIRST(β) contains ε, then everything in FOLLOW(A) is in FOLLOW(B).

Code :

```cpp
#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("input.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;
}
```

Manual Calculation :



Given Grammar is

$S \rightarrow ACB \mid CbB \mid Ba$

$A \rightarrow da \mid BC$

$B \rightarrow g \mid \epsilon$

$C \rightarrow h \mid \epsilon$

First: (just see what are the terminals in the Production rules which comes first)

First(S) = {d, g, h, ∈, b, a}

First(a) = {d, g, h, ∈}

First(B) = {g, ∈}

First(C) = {h, ∈}

Follow:

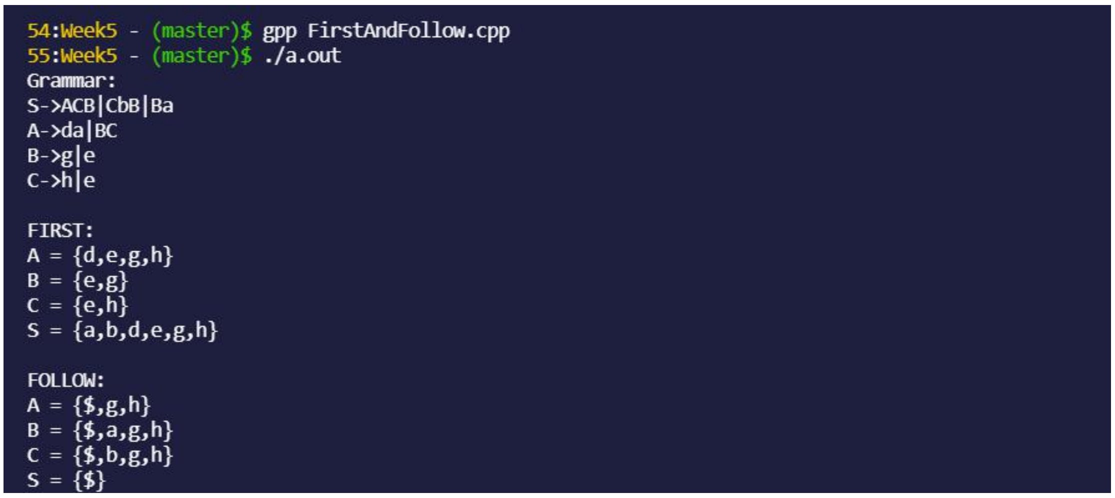
follow(S) = {$}

follow(A) = {h, g, $}

Follow(B) = {$, a, h, g}

Follow(C) = {g, $, b, h}

Output :


```
54:Week5 - (master)$ gpp FirstAndFollow.cpp
55:Week5 - (master)$ ./a.out
Grammar:
S->ACB|CbB|Ba
A->da|BC
B->g|e
C->h|e

FIRST:
A = {d,e,g,h}
B = {e,g}
C = {e,h}
S = {a,b,d,e,g,h}

FOLLOW:
A = {$,g,h}
B = {$,a,g,h}
C = {$,b,g,h}
S = {$}
```

Result:

First and Follow of the given grammar is computed successfully implemented and verified.