Program 5:

WAP to Evaluate the FIRST & FOLLOW information of a CFG which is given through a file.

```
#include <iostream>
#include <vector>
#include <map>
#include <set>
#include <fstream>
#include <algorithm>

using namespace std;

template <typename T>

int Read(char *str)
{
 int i = 0;
 while (1)
 {
  str[i] = getchar();
  if (str[i] == '\n' || str[i] == '\r')
  {
   str[i] = '\0';
   return i;
  }
  i++;
 }
}

void printSet(set<char> &set1)
{
 for (auto i = set1.begin(); i != set1.end(); ++i)
  cout << *i << " ";
 cout << endl;
}

set<char>::iterator intersection(set<char> &set1, set<char> &set2)
{
 for (auto i = set1.begin(); i != set1.end(); i++)
```

```
  for (auto j = set2.begin(); j != set2.end(); j++)
   if ((*i) == (*j))
    return i;
 return set1.end();
}

int main()
{
 char ch1;

 vector<vector<char>> CFG2D;
 vector<char> tempVector;
 set<char> nonTerminals;
 set<char> terminals;
 set<char> tempSet;
 map<char, set<char>> firstOf;
 map<char, set<char>> followOf;

 FILE *filePointer = NULL;
 filePointer = fopen("FirstFollow4.txt", "r");
 if (filePointer == NULL)
 {
  printf("Unable to open FirstFollow.txt\n");
  return 1;
 }

 char startSymbol;
 fscanf(filePointer, "%c%c", &startSymbol, &ch1);
 printf("Start symbol : %c\n", startSymbol);

 while (fscanf(filePointer, "%c", &ch1) != EOF)
 {
  if (ch1 == '\n')
  {
   CFG2D.push_back(tempVector);
```

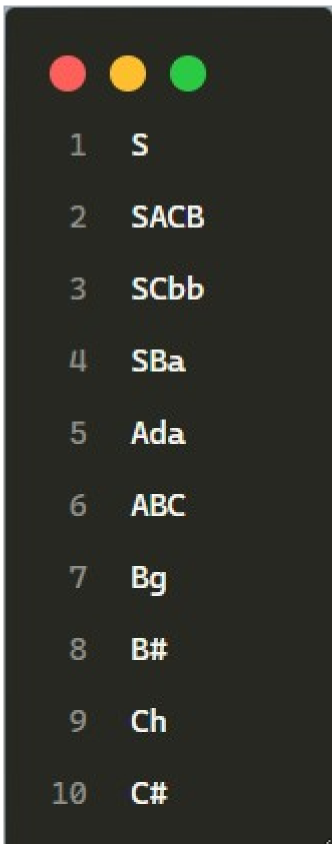
```cpp
  tempVector.clear();
  continue;
 }
 tempVector.push_back(ch1);
 if (ch1 >= 'A' && ch1 <= 'Z')
  nonTerminals.insert(ch1);
 else if (ch1 == '#')
  continue;
 else
  terminals.insert(ch1);
}
CFG2D.push_back(tempVector);

fclose(filePointer);

cout << "Terminals : ";
for (auto terminal : terminals)
 cout << terminal << " ";
cout << endl;
cout << "Non Terminals : ";
for (auto nonTerminal : nonTerminals)
 cout << nonTerminal << " ";
cout << endl;

cout << "CFG : \n";
for (auto i = CFG2D.begin(); i != CFG2D.end(); ++i)
{
 auto j = (*i).begin();
 cout << *j << " --> ";
 ++j;
 for (; j != (*i).end(); ++j)
  cout << *j;
 cout << endl;
}
cout << endl;
```

```
//* caluclate first
for (auto i = CFG2D.begin(); i != CFG2D.end(); ++i)
{
 auto j = (*i).begin();
 ch1 = *j;
 ++j;
 firstOf[ch1].insert(*j);
}

int flag = 0;

do
{
 flag = 0;
 for (auto i = CFG2D.begin(); i != CFG2D.end(); ++i)
 {
  auto j = (*i).begin();
  char currentTerminal = *j;
  ++j;
  char firstOfNonTerminal = *j;

  // if firstOfNonTerminal is a terminal
  if (nonTerminals.find(firstOfNonTerminal) == nonTerminals.end())
   continue;

  // if first of RHS nonTerminal has notTerminal then recheck and skip current 
LHS
  set<char>::iterator it = intersection(firstOf[firstOfNonTerminal], 
nonTerminals);
  if (firstOf[firstOfNonTerminal].end() != it)
  {
   flag = 1;
   continue;
  }
```

```cpp
   // if first of RHS nonTerminal has no '#' copy first then continue
   if (firstOf[firstOfNonTerminal].find('#') == firstOf[firstOfNonTerminal].end())
   {
    firstOf[currentTerminal].erase(firstOfNonTerminal);
    firstOf[currentTerminal].insert(firstOf[firstOfNonTerminal].begin(),
firstOf[firstOfNonTerminal].end());
    continue;
   }

   // if first of RHS nonTerminal has '#'
   while (++j != (*i).end())
   {
    bool containedEpsilonAlready = firstOf[currentTerminal].find('#') !=
firstOf[currentTerminal].end();
    if (terminals.find(*j) != terminals.end())
    {
     firstOf[currentTerminal].erase(firstOfNonTerminal);
     firstOf[currentTerminal].insert(firstOf[firstOfNonTerminal].begin(),
firstOf[firstOfNonTerminal].end());
     firstOf[currentTerminal].insert(*j);
     firstOf[currentTerminal].erase('#');
     if (containedEpsilonAlready)
      firstOf[currentTerminal].insert('#');
     break;
    }
    firstOf[currentTerminal].erase(firstOfNonTerminal);
    firstOf[currentTerminal].insert(firstOf[firstOfNonTerminal].begin(),
firstOf[firstOfNonTerminal].end());
    firstOf[currentTerminal].erase('#');
    if (containedEpsilonAlready)
     firstOf[currentTerminal].insert('#');

    firstOfNonTerminal = *j;
    set<char>::iterator it = intersection(firstOf[firstOfNonTerminal],
nonTerminals);
    if (it != firstOf[firstOfNonTerminal].end())
    {
     flag = 1;
     break;
    }

    if (firstOf[firstOfNonTerminal].find('#') == 
firstOf[firstOfNonTerminal].end())
    {
     firstOf[currentTerminal].erase(firstOfNonTerminal);
     firstOf[currentTerminal].insert(firstOf[firstOfNonTerminal].begin(), 
firstOf[firstOfNonTerminal].end());
     break;
    }
   }
   if (j == (*i).end())
   {
    firstOf[currentTerminal].erase(firstOfNonTerminal);
    firstOf[currentTerminal]
      .insert(firstOf[firstOfNonTerminal].begin(), 
firstOf[firstOfNonTerminal].end());
    firstOf[currentTerminal].insert('#');
    continue;
   }
  }
 } while (flag);

 //* calculating follow
 followOf[startSymbol].insert('$');

 for (auto nonTerminal = nonTerminals.begin(); nonTerminal != nonTerminals.end(); 
++nonTerminal)
 {
  for (auto i = CFG2D.begin(); i != CFG2D.end(); ++i)
```

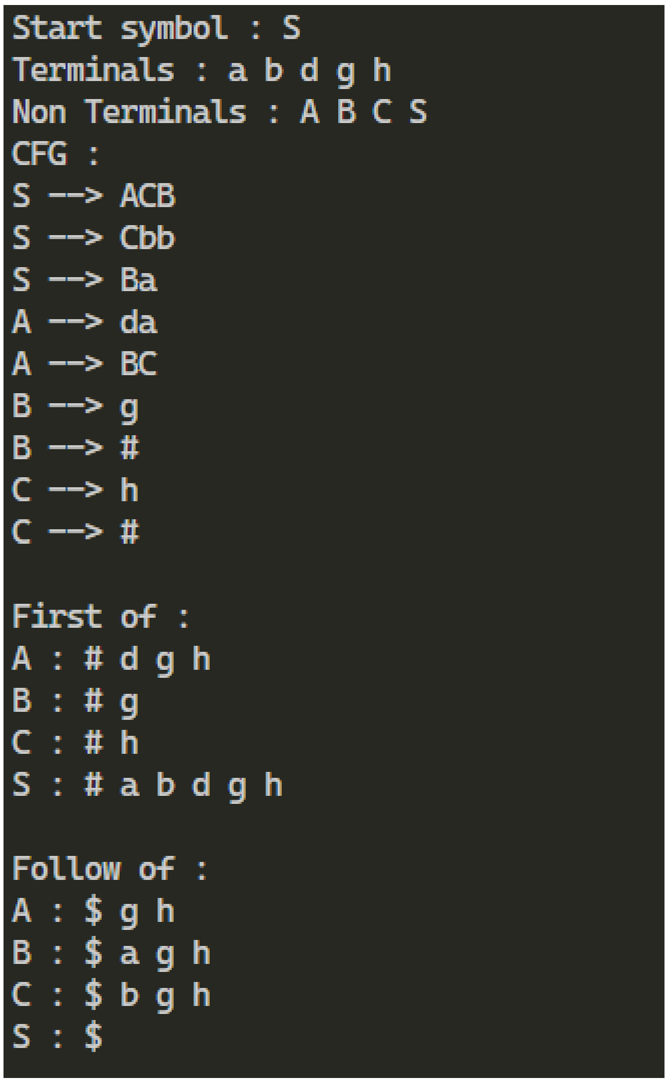
```cpp
  {
   auto j = (*i).begin();
   char producingNonTerminal = *j;
   while (++j != (*i).end())
   {
    char firstOfNonTerminal = *j;
    if (firstOfNonTerminal == *nonTerminal)
    {
     auto next = j + 1;
     if (next == (*i).end())
     {
      followOf[*nonTerminal].insert(producingNonTerminal);
      break;
     }
     else if (terminals.find(*next) != terminals.end())
     {
      followOf[*nonTerminal].insert(*next);
      break;
     }
     else
     {
      while (next != (*i).end())
      {
       if (terminals.find(*next) != terminals.end())
       {
        followOf[*nonTerminal].insert(*next);
        break;
       }
       if (firstOf[*next].find('#') != firstOf[*next].end())
       {
        followOf[*nonTerminal].insert(firstOf[*next].begin(),
firstOf[*next].end());
        followOf[*nonTerminal].erase('#');
        ++next;
       }
       else
       {
        followOf[*nonTerminal].insert(firstOf[*next].begin(),
firstOf[*next].end());
        break;
       }
      }
      if (next == (*i).end())
       followOf[*nonTerminal].insert(producingNonTerminal);
      break;
     }
    }
   }
  }
 }

 flag = 1;
 while (flag)
 {
  flag = 0;
  for (auto nonTerminal = nonTerminals.begin(); nonTerminal != nonTerminals.end();
++nonTerminal)
  {
   for (auto followTerminal = followOf[*nonTerminal].begin();
followOf[*nonTerminal].end() != followTerminal; ++followTerminal)
   {
    if (nonTerminals.find(*followTerminal) != nonTerminals.end())
    {
     flag = 1;
     if (followTerminal == nonTerminal)
      followOf[*nonTerminal].erase(*nonTerminal);
     else
     {
      followOf[*nonTerminal].erase(*followTerminal);
      followOf[*nonTerminal].insert(followOf[*followTerminal].begin(),
followOf[*followTerminal].end());
     }
    }
   }
  }
 }

 //* Print first of
 cout << "First of : \n";
 for (auto i = firstOf.begin(); i != firstOf.end(); ++i)
 {
  ch1 = (*i).first;
  printf("%c : ", ch1);
  auto &firstOfCh = firstOf[ch1];
  printSet(firstOfCh);
 }
 cout << endl;

 //* Print first of
 cout << "Follow of : \n";
 for (auto i = followOf.begin(); i != followOf.end(); ++i)
 {
  ch1 = (*i).first;
  printf("%c : ", ch1);
  auto &followOfCh = followOf[ch1];
  printSet(followOfCh);
 }
 cout << endl;

 return 0;
}
```

<u>FirstFollow4.txt</u>



```
S
SACB
SCbb
SBa
Ada
ABC
Bg
B#
Ch
C#
```

Output:



```
Start symbol : S
Terminals : a b d g h
Non Terminals : A B C S
CFG :
S --> ACB
S --> Cbb
S --> Ba
A --> da
A --> BC
B --> g
B --> #
C --> h
C --> #

First of :
A : # d g h
B : # g
C : # h
S : # a b d g h

Follow of :
A : $ g h
B : $ a g h
C : $ b g h
S : $
```