Lab-07

Name: Bhalodia Akshar

Roll No.: IT008

Aim: WAP to find first and follow sets, of each NT and each grammar rule.

Code:

```
#include<bits/stdc++.h>
using namespace std;

bool comp(string &s1, string &s2) {
   return s1.size() > s2.size();
}
string filter(string str) {
   for (int i = 0; i < str.length(); i += 3) {
      for (int j = i + 3; j < str.length(); j += 3) {
         if (str.substr(i, 3) == str.substr(j, 3)) {
            str = str.substr(0, j) + str.substr(j + 3);
         }
      }
   }
   return str;
}

string filterNull(string str) {
  str = filter(str);
  for (int i = 0; i < str.length(); i += 3) {
```

```
    if (str.substr(i, 3) == "^, ")
      str = str.substr(0, i) + str.substr(i + 3);
  }
  return str;
}

void find() {
  string input[100][100], NT[100];
  int proCount[100];
  int expCount;
  cout << "Enter number of expression: ";
  cin >> expCount;
  for (int i = 0; i < expCount; i++) {
    cout << "\nEnter non-terminal: ";
    cin >> NT[i];
    cout << "\nEnter number of productions: ";
    cin >> proCount[i];
    int j = 0;
    for (j = 0; j < proCount[i]; j++) {
      cout << "\nEnter production " << j + 1 << " : ";
      cin >> input[i][j];
    }
    sort(input[i], input[i] + j, comp);
  }
  cout << "\nGiven input Expression is:\n";
  for (int i = 0; i < expCount; i++) {
    cout << NT[i] << " -> ";
    cout << input[i][0];
    for (int j = 1; j < proCount[i]; j++) {
      cout << " | " << input[i][j];
```

```
  }
  cout << endl;
}
string first[expCount];
for (int dm = 0; dm < expCount; ++dm) {
  for (int i = 0; i < expCount; ++i) {
    for (int j = 0; j < proCount[i]; j++) {
      /// check next is ^ or not
      int m = 0;
      for (; m < input[i][j].length(); ++m) {
        // check terminal or non terminal
        int nonTerIndex = -1;
        for (int n = 0; n < expCount; ++n) {
          if (input[i][j].substr(m, NT[n].length()) == NT[n]) {
            nonTerIndex = n;
            break;
          }
        }
        if (nonTerIndex == -1) {
          first[i] = filter(first[i] + input[i][j].substr(m, 1) + ", ");
          break;
        }
        if (first[nonTerIndex].find('^') != -1) {
          first[i] = filter(first[i] + first[nonTerIndex]);
        } else {
          first[i] = filter(first[i] + first[nonTerIndex]);
          break;
        }
      }
    }
```

```
  }
}
cout << "\nFirst set of Given Expression is:\n";
for (int i = 0; i < expCount; i++) {
  cout << NT[i] << " -> ";
  cout << first[i];
  cout << endl;
}
string follow[expCount];
cout << follow[1];
if (expCount)
  follow[0] = "$, ";
for (int dm = 0; dm < expCount; ++dm) {
  for (int i = 0; i < expCount; ++i) {
    for (int j = 0; j < expCount; ++j) {
      for (int k = 0; k < proCount[j]; ++k) {
        for (int l = 0; l < input[j][k].length(); ++l) {
          if (input[j][k].substr(l, NT[i].length()) == NT[i]) {
            string nextString = input[j][k].substr(l + NT[i].length());
            if (nextString.length() == 0) {
              follow[i] = filterNull(follow[i] + follow[j]);
              continue;
            } else {
              int m = 0;
              for (; m < nextString.length(); ++m) {
                int nonTerIndex = -1;
                for (int n = 0; n < expCount; ++n) {
                  if (nextString.substr(m, NT[n].length()) == NT[n]) {
                    nonTerIndex = n;
                    break;
```

```
            }
          }
          if (nonTerIndex == -1) {
            follow[i] = filterNull(follow[i] + nextString.substr(m, 1) + ", ");
            break;
          }
          if (first[nonTerIndex].find('^') != -1) {
            follow[i] = filterNull(follow[i] + first[nonTerIndex]);
            continue;
          } else {
            follow[i] = filterNull(follow[i] + first[nonTerIndex]);
            break;
          }
        }
        if (m == nextString.length()) {
          follow[i] = filterNull(follow[i] + follow[j]);
        }
      }
    }
  }
 }
}
cout << "\nFollow of Given Expression is:\n";
for (int i = 0; i < expCount; i++) {
  cout << NT[i] << " -> ";
  cout << follow[i];
  cout << endl;
}
```

```
}
int main() {
  int i = 1;
  while (i) {
    find();
    i--;
  }
  return 0;
}
```

Output: