Practical 3

Niketkumar Kothari
18BCE134

Aim: To Find the First() and Follow() of Grammar

Solution: -

```
#include <map>
#include <set>
#include <string>
#include <vector>
 using namespace std;
int terminalCount;
int nonTerminalCount;
int productionCount;
 map<string, vector<string>> ps_map;
map<string, set<string>> FIRST;
map<string, set<string>> FOLLOW;
 string *ts;
string *non_ts;
string ss;
string *ps;
 template <typename T> set<T> getUnion(const set<T> &a, const set<T> &b) {
set<T> result = a;
  result.insert(b.begin(), b.end());
return result;
} void c_p_c() { #ifndef
ONLINE_JUDGE   freopen("input.txt",
"r", stdin);   freopen("o.txt", "w",
stdout);
#endif
}  string getString(char x)
{   string s(1, x);  return
s;
}
vector<string> sp(string input, string delimiter) {
size_t pos = 0;   string token;   vector<string> prods;
```

```
while ((pos = input.find(delimiter)) != string::npos) {
token = input.substr(0, pos);
prods.push_back(token);     input.erase(0, pos +
delimiter.length());
  }
prods.push_back(input);
return prods;
}
       set<string> ne = first_2;
  ne.erase("@");
bool inArray(string s, string *array, int size) {
for (int i = 0; i < size; i++) {     if (array[i]
== s)       return true;
  }   return
false;
}  set<string> first(string s)
{   using namespace std;
   set<string>
first_;
   if (inArray(s, non_ts, nonTerminalCount))
{     vector<string> alternatives =
ps_map[s];
     for (int i = 0; i < alternatives.size(); ++i) {
string temp = alternatives[i];       set<string>
first_2 = first(temp);       first_ =
getUnion(first_, first_2);
    }

  } else if (inArray(s, ts, terminalCount)) {
first_ = {s};
  } else if (s == "" || s == "@") {
first_ = {"@"};
  } else {     set<string> first_2 =
first(getString(s[0]));     if (first_2.find("@")
!= first_2.end()) {       int i = 1;     while
(first_2.find("@") != first_2.end()) {
       first_ = getUnion(first_, ne);

        if (inArray(s.substr(i), ts, terminalCount)) {
  set<string> t = {s.substr(i)};
```

```
          first_ = getUnion(first_, t);
  break;
        } else if (s.substr(i) == "") {
set<string> t = {"@"};           first_
= getUnion(first_, t);           break;
        }          ne =
first(s.substr(i));           if
(nT == nt) {             continue;
        ne.erase("@");         first_ =
getUnion(first_, ne);         i++;
}     } else {       first_ =
getUnion(first_, first_2);
    }   }
return first_;
}  set<string> follow(string nT)
{   using namespace std;
   set<string>
follow_;
   if (nT == ss) {     set<string> dollar
= {"$"};     follow_ = getUnion(follow_,
dollar);
  }    map<string, vector<string>>::iterator itr;   for
(itr = ps_map.begin(); itr != ps_map.end(); ++itr) {
string nt = itr->first;     vector<string> rhs = itr-
>second;
     for (auto alt = rhs.begin(); alt != rhs.end(); ++alt) {
for (int i = 0; i < (*alt).length(); i++) {         if (nT
== getString((*alt)[i])) {           string following_str =
(*alt).substr(i + 1);         if (following_str == "") {
          } else {              follow_ =
 getUnion(follow_, follow(nt));
            }
          } else {            set<string> follow_2 =
 first(following_str);
            if (follow_2.find("@") != follow_2.end()) {
              set<string> t = follow_2;
```

```
            t.erase("@");                follow_ =
getUnion(follow_, t);               follow_ =
getUnion(follow_, follow(nt));
           } else {                follow_ =
getUnion(follow_, follow_2);
           }
         }
       }
     }
   }   }   return
follow_;
}  int main() {   c_p_c();
cout << "Enter no. of ts: ";
cin >> terminalCount;
   ts = new
string[terminalCount];
   cout << "Enter the ts :" << endl;   for
(int i = 0; i < terminalCount; i++) {
cin >> ts[i];
  }

  // Non ts
   cout << "Enter no. of non ts:
";   cin >> nonTerminalCount;
   non_ts = new string[nonTerminalCount];
   cout << "Enter the non ts :" << endl;   for
(int i = 0; i < nonTerminalCount; i++) {
cin >> non_ts[i];
  }
  cout << "Enter the starting symbol: ";
```

```
  cin >> ss;

  cout << "Enter the number of ps: ";
  cin >> productionCount; ps = new
  string[productionCount];


  cout << "Enter the ps: ";
   for (int i = 0; i < productionCount; i++) {
cin >> ps[i];     vector<string> temp =
sp(ps[i], "->");     vector<string> temp2 =
sp(temp[1], "|");
    ps_map.insert(pair<string, vector<string>>(temp[0], temp2));
}    map<string, vector<string>>::iterator itr;   for (itr =
ps_map.begin(); itr != ps_map.end(); ++itr) {     cout << itr-
>first << " -> ";     vector<string>::iterator i;     for (i =
itr->second.begin(); i != itr->second.end(); ++i) {       cout
<< *i << " ";
    }     cout <<
endl;
  }
  for (int i = 0; i < nonTerminalCount; i++) {
    FIRST[non_ts[i]] = {};
    FOLLOW[non_ts[i]] = {};
  }
  for (int i = 0; i < nonTerminalCount; i++) {
    FIRST[non_ts[i]] = getUnion(FIRST[non_ts[i]], first(non_ts[i]));
  }
  set<string> dollar = {"$"};
  FOLLOW[ss] = getUnion(FOLLOW[ss], dollar);

  for (int i = 0; i < nonTerminalCount; i++) {
    FOLLOW[non_ts[i]] = getUnion(FOLLOW[non_ts[i]], follow(non_ts[i]));
```

```
  }    cout << "Non ts \t First \t\tFollow" << endl;   for (int i = 0; i <
nonTerminalCount; i++) {     cout << non_ts[i] << " \t\t ";     for (auto
itr = FIRST[non_ts[i]].begin(); itr != FIRST[non_ts[i]].end();
      ++itr) {      cout
 << *itr << " ";
    }   cout <<
  "\t\t";
    for (auto itr = FOLLOW[non_ts[i]].begin(); itr !=
 FOLLOW[non_ts[i]].end();
         ++itr) {
cout << *itr << " ";
    }     cout <<
endl;
  }
return 0;
}
```

Execution: -

```
Enter no. of terminals: 3
Enter the terminals :
a
b
c
Enter no. of non terminals: 2
Enter the non terminals :
S
A
Enter the starting symbol: S
Enter the number of productions: 2
Enter the productions: S->Aa
A->bc
A -> bc
S -> Aa
Non Terminals    First          Follow
S                b              $
A                b              a
```