**Name: Sachin Vishwakarma Subject:Compiler Roll no:537**

**First: FIRST ()**− **It is a function that gives the set of terminals that begin the strings derived from the production rule. A symbol c is in FIRST (α) if and only if α** ⇒ **cβ for some sequence β of grammar symbols**

**Follow:A terminal symbol a is in FOLLOW (N) if and only if there is a derivation from the start symbol S of the grammar such that S** ⇒ **αNαβ, where α and β are a (possible empty) sequence of grammar symbols. In other words, a terminal c is in FOLLOW (N) if c can follow N at some point in a derivation.**

**Aim: Find first and follow.**

import re

def cal\_follow(s, productions, first):

follow = set()

if len(s)!=1 :

return {}

if(s == list(productions.keys())[0]):

follow.add('$')

for i in productions:

for j in range(len(productions[i])):

if(s in productions[i][j]):

idx = productions[i][j].index(s)

if(idx == len(productions[i][j])-1):

if(productions[i][j][idx] == i):

break

else:

f = cal\_follow(i, productions, first) for x in f:

follow.add(x)

else:

while(idx != len(productions[i][j]) - 1): idx += 1

if(not productions[i][j][idx].isupper()): follow.add(productions[i][j][idx]) break

else:

f = cal\_first(productions[i][j][idx], productions)

if('ε' not in f):

for x in f:

follow.add(x)

break

elif('ε' in f and idx != len(productions\*i+\*j+)-1):

f.remove('ε')

for k in f:

follow.add(k)

elif('ε' in f and idx == len(productions[i][j])-1): f.remove('ε')

for k in f:

follow.add(k)

f = cal\_follow(i, productions, first) for x in f:

follow.add(x)

return follow

def cal\_first(s, productions):

first = set()

for i in range(len(productions[s])):

for j in range(len(productions[s][i])):

c = productions[s][i][j]

if(c.isupper()):

f = cal\_first(c, productions) if('ε' not in f):

for k in f:

first.add(k)

break

else:

if(j == len(productions[s][i])-1): for k in f:

first.add(k)

else:

f.remove('ε')

for k in f:

first.add(k)

else:

first.add(c)

break

return first

def main():

productions = {}

grammar = open("grammer9.txt", "r")

first = {}

follow = {}

for prod in grammar:

l = re.split("( /->/\n/)\*", prod)

m = []

for i in l:

if (i == "" or i == None or i == '\n' or i == " " or i == "-" or i == ">"): pass

else:

m.append(i)

left\_prod = m.pop(0)

right\_prod = []

t = []

for j in m:

if(j != '|'):

t.append(j)

else:

right\_prod.append(t)

t = []

right\_prod.append(t)

productions[left\_prod] = right\_prod

for s in productions.keys():

first[s] = cal\_first(s, productions)

print("\*\*\*\*\*FIRST\*\*\*\*\*")

for lhs, rhs in first.items():

print(lhs, ":" , rhs)

print("")

for lhs in productions:

follow[lhs] = set()

for s in productions.keys():

follow[s] = cal\_follow(s, productions, first)

print("\*\*\*\*\*FOLLOW\*\*\*\*\*")

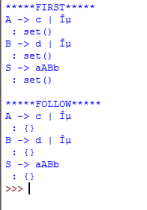
for lhs, rhs in follow.items():

print(lhs, ":" , rhs)

grammar.close()

if \_\_name\_\_ == "\_\_main\_\_": main()

**Output:**

****