Name: Adwait Hegde
Roll No: 2019130019
TE Comp (Batch-A)

EXPERIMENT 3

Aim:

Given a grammar find the first and follow set of non-terminals.

Theory:

FIRST SET

FIRST(X) for a grammar symbol X is the set of terminals that begin the strings derivable from X.

Rules to compute FIRST set:

1. If x is a terminal, then FIRST(x) = { ‘x’ }
2. If x-> Є, is a production rule, then add Є to FIRST(x).
3. If X->Y1 Y2 Y3….Yn is a production,
    1. FIRST(X) = FIRST(Y1)
    2. If FIRST(Y1) contains Є then FIRST(X) = { FIRST(Y1) – Є } U { FIRST(Y2) }
    3. If FIRST (Yi) contains Є for all i = 1 to n, then add Є to FIRST(X).

Example:

Production Rules of Grammar
S -> ACB | Cbb | Ba
A -> da | BC
B -> g | Є
C -> h | Є

FIRST sets

FIRST(S) = FIRST(ACB) U FIRST(Cbb) U FIRST(Ba)
= { d, g, h, b, a, Є}
FIRST(A) = { d } U FIRST(BC)
= { d, g, h, Є }
FIRST(B) = { g , Є }
FIRST(C) = { h , Є }

1. The grammar used above is Context-Free Grammar (CFG). Syntax of most of the programming language can be specified using CFG.
2. CFG is of the form A -> B , where A is a single Non-Terminal, and B can be a set of grammar symbols (i.e. Terminals as well as Non-Terminals)

FOLLOW SET

Follow(X) to be the set of terminals that can appear immediately to the right of Non-Terminal X in some sentential form.

Rules to compute FOLLOW set:

1. FOLLOW(S) = { $ }   // where S is the starting Non-Terminal
2. If A -> pBq is a production, where p, B and q are any grammar symbols, then everything in FIRST(q) except Є is in FOLLOW(B).
3. If A->pB is a production, then everything in FOLLOW(A) is in FOLLOW(B).
4. If A->pBq is a production and FIRST(q) contains Є, then FOLLOW(B) contains { FIRST(q) – Є } U FOLLOW(A)

Example:

Production Rules:

E -> TE’
E’ -> +T E’|Є
T -> F T’
T’ -> *F T’ | Є
F -> (E) | id

FIRST set

FIRST(E) = FIRST(T) = { (, id }
FIRST(E’) = { +, Є }
FIRST(T) = FIRST(F) = { (, id }
FIRST(T’) = { *, Є }
FIRST(F) = { (, id }

FOLLOW Set

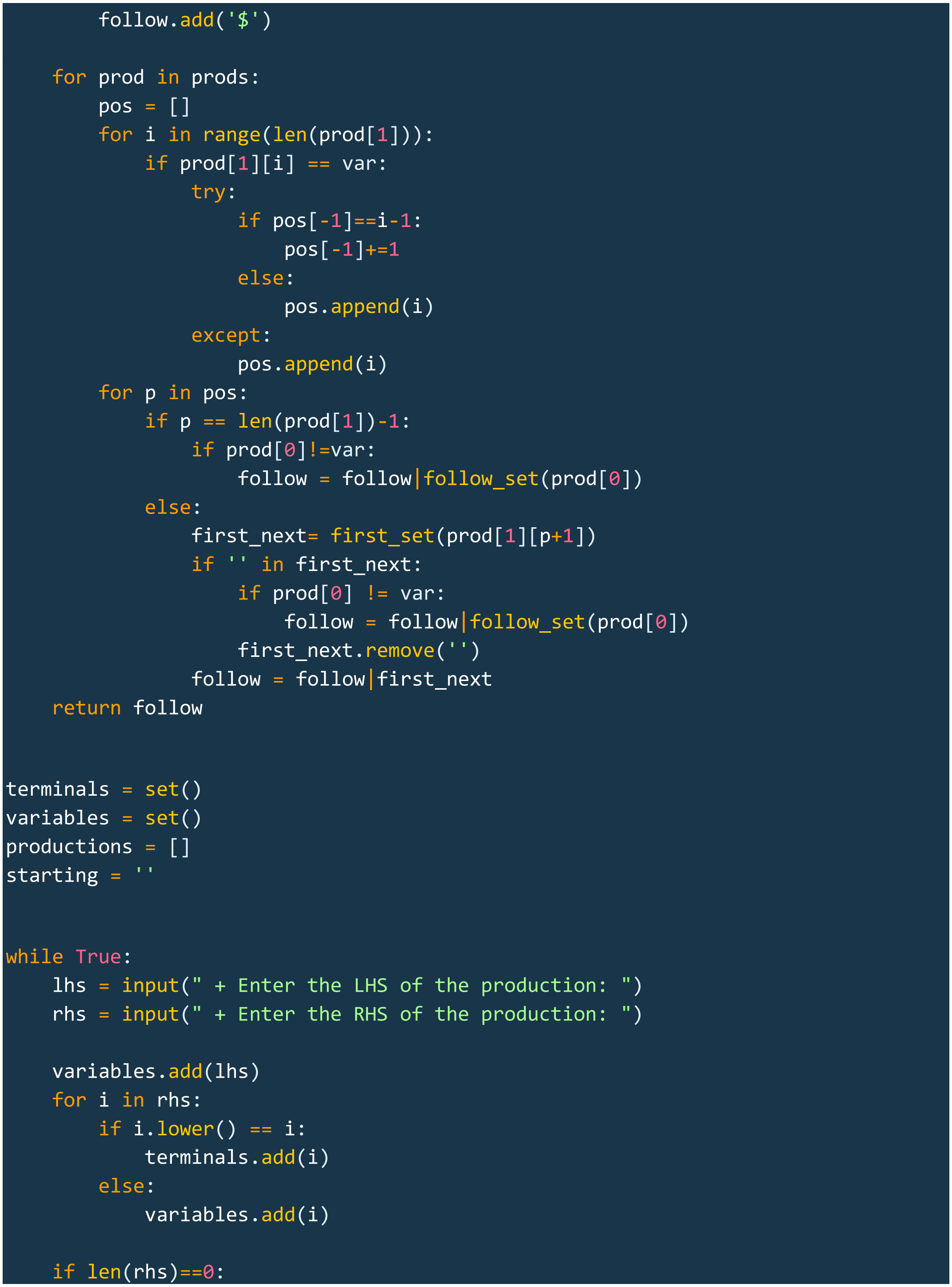
FOLLOW(E) = { $,) } // Note ')' is there because of 5th rule
FOLLOW(E’) = FOLLOW(E) = { $,) } // See 1st production rule
FOLLOW(T) = { FIRST(E’) – Є } U FOLLOW(E’) U FOLLOW(E) = { + , $,) }
FOLLOW(T’) = FOLLOW(T) =   { + , $,) }
FOLLOW(F) = { FIRST(T’) – Є } U FOLLOW(T’) U FOLLOW(T) = { *, +, $,) }

1. Є as a FOLLOW doesn’t mean anything (Є is an empty string).
2. $ is called end-marker, which represents the end of the input string, hence used while parsing to indicate that the input string has been completely processed.
3. The grammar used above is Context-Free Grammar (CFG). The syntax of a programming language can be specified using CFG.
4. CFG is of the form A -> B , where A is a single Non-Terminal, and B can be a set of grammar symbols (i.e. Terminals as well as Non-Terminals)

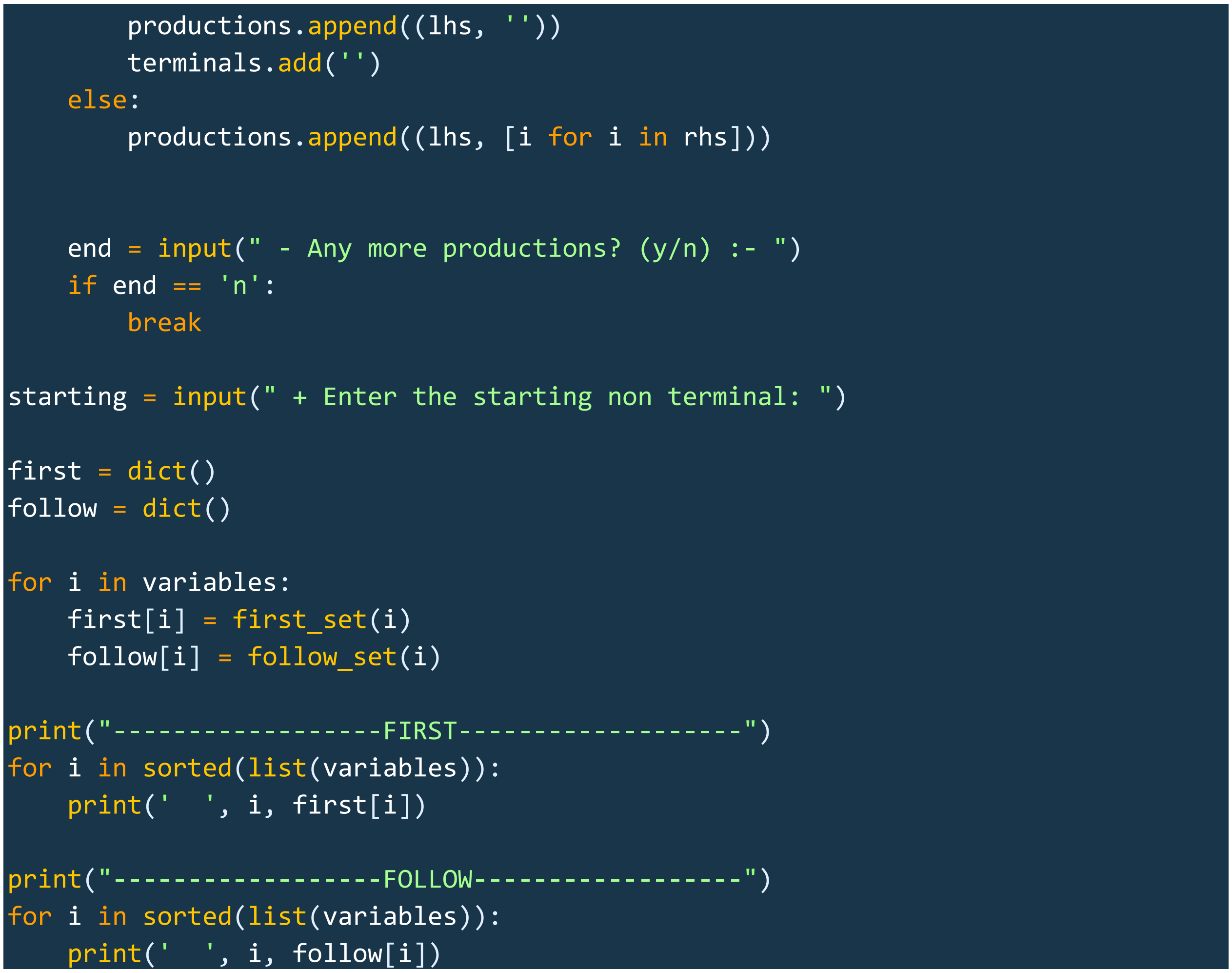
Implementation

Code:

```
def first_set(var):
    global productions, variables, terminals
    first = set()
    prods = []
    if var in terminals:
        return {var}

    for production in productions:
        if production[0] == var:
            prods.append(production[1])
    for prod in prods:
        if len(prod)==0:
            first.add('')
        elif prod[0] in terminals:
            first.add(prod[0])
        else:
            c = 0
            flag = True
            while flag:
                first_var = first_set(prod[c])
                if '' in first_var:
                    first_var.remove('')
                else:
                    flag=False

                first = first|first_var
                c+=1
                if c== len(prod) and flag:
                    first.add('')
                    flag = False
    return first


def follow_set(var):
    global productions, variables, terminals, starting
    prods = []
    for production in productions:
        if var in production[1]:
            prods.append(production)
    follow = set()

    if var == starting:
```



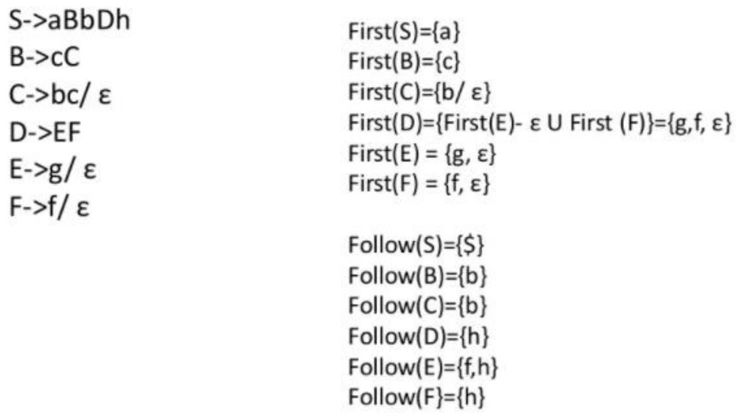
```
        follow.add('$')

    for prod in prods:
        pos = []
        for i in range(len(prod[1])):
            if prod[1][i] == var:
                try:
                    if pos[-1]==i-1:
                        pos[-1]+=1
                    else:
                        pos.append(i)
                except:
                    pos.append(i)
        for p in pos:
            if p == len(prod[1])-1:
                if prod[0]!=var:
                    follow = follow|follow_set(prod[0])
            else:
                first_next= first_set(prod[1][p+1])
                if '' in first_next:
                    if prod[0] != var:
                        follow = follow|follow_set(prod[0])
                    first_next.remove('')
                follow = follow|first_next
    return follow


terminals = set()
variables = set()
productions = []
starting = ''


while True:
    lhs = input(" + Enter the LHS of the production: ")
    rhs = input(" + Enter the RHS of the production: ")

    variables.add(lhs)
    for i in rhs:
        if i.lower() == i:
            terminals.add(i)
        else:
            variables.add(i)

    if len(rhs)==0:
```



```
        productions.append((lhs, ''))
        terminals.add('')
    else:
        productions.append((lhs, [i for i in rhs]))


    end = input(" - Any more productions? (y/n) :- ")
    if end == 'n':
        break

starting = input(" + Enter the starting non terminal: ")

first = dict()
follow = dict()

for i in variables:
    first[i] = first_set(i)
    follow[i] = follow_set(i)

print("------------------FIRST-------------------")
for i in sorted(list(variables)):
    print('  ', i, first[i])

print("------------------FOLLOW------------------")
for i in sorted(list(variables)):
    print('  ', i, follow[i])
```

Result:

Explanation:

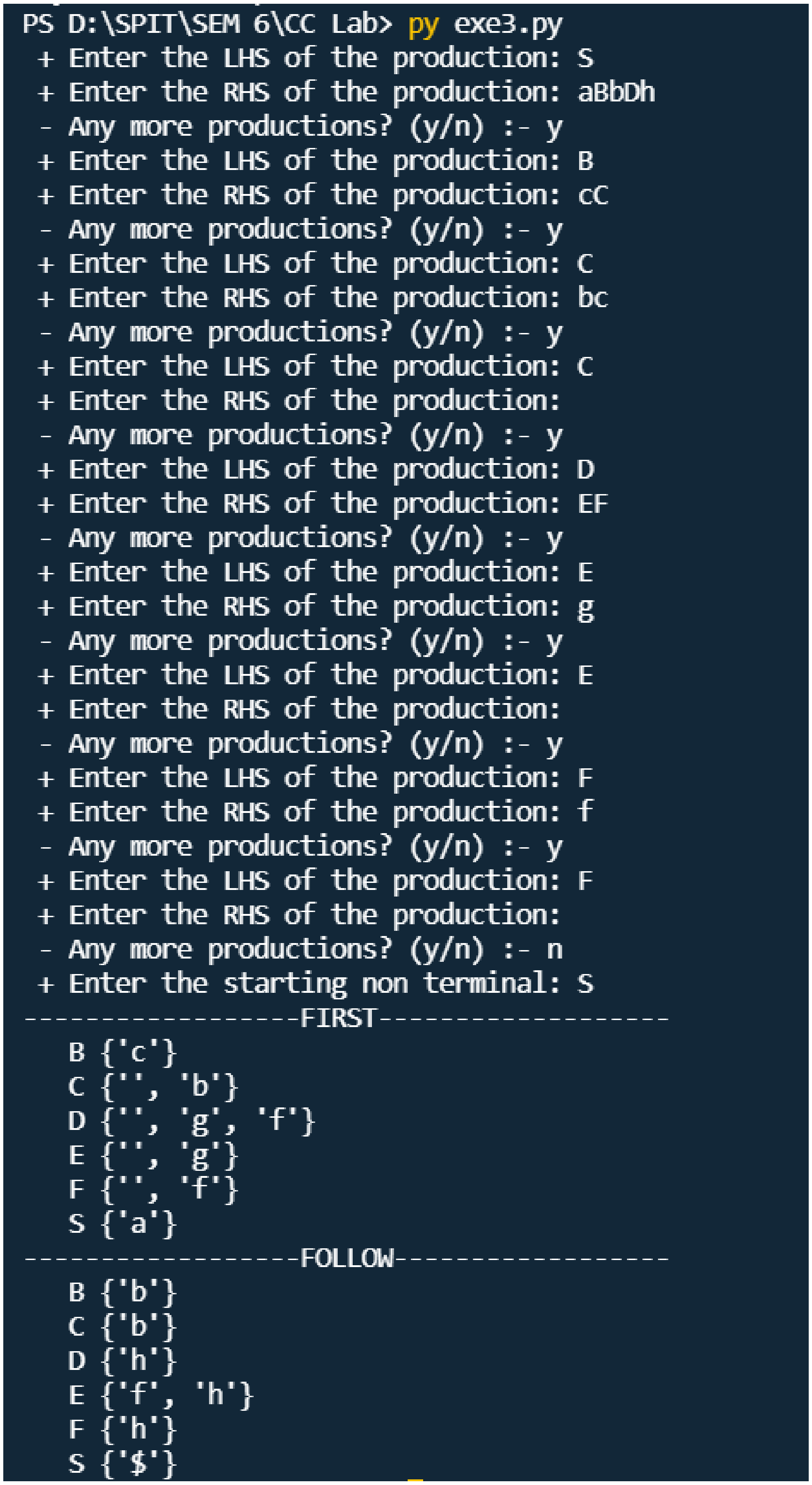


S->aBbDh
B->cC
C->bc/ ε
D->EF
E->g/ ε
F->f/ ε

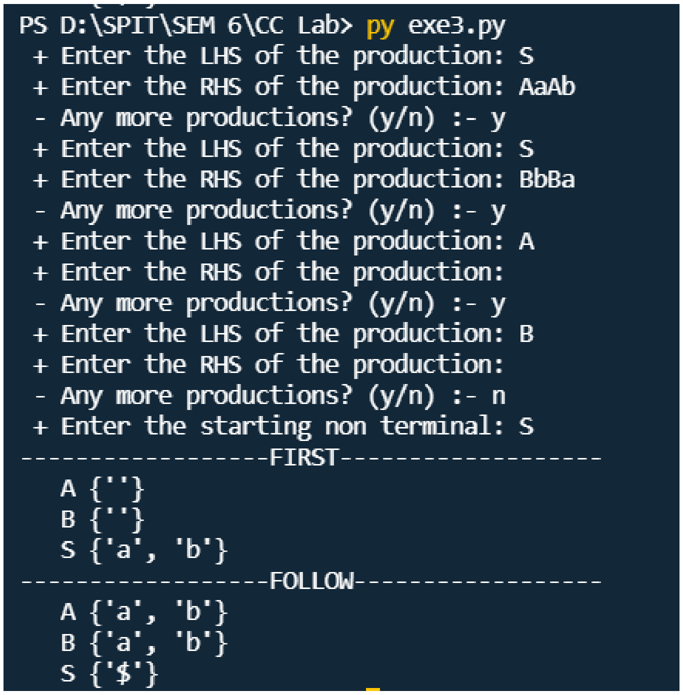
First(S)={a}
First(B)={c}
First(C)={b/ ε}
First(D)={First(E)- ε U First (F)}={g,f, ε}
First(E) = {g, ε}
First(F) = {f, ε}

Follow(S)={$}
Follow(B)={b}
Follow(C)={b}
Follow(D)={h}
Follow(E)={f,h}
Follow(F)={h}

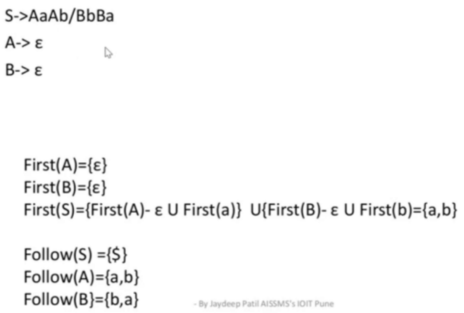
Output: (considering ‘ ’ as Є)



```
PS D:\SPIT\SEM 6\CC Lab> py exe3.py
 + Enter the LHS of the production: S
 + Enter the RHS of the production: aBbDh
 - Any more productions? (y/n) :- y
 + Enter the LHS of the production: B
 + Enter the RHS of the production: cC
 - Any more productions? (y/n) :- y
 + Enter the LHS of the production: C
 + Enter the RHS of the production: bc
 - Any more productions? (y/n) :- y
 + Enter the LHS of the production: C
 + Enter the RHS of the production:
 - Any more productions? (y/n) :- y
 + Enter the LHS of the production: D
 + Enter the RHS of the production: EF
 - Any more productions? (y/n) :- y
 + Enter the LHS of the production: E
 + Enter the RHS of the production: g
 - Any more productions? (y/n) :- y
 + Enter the LHS of the production: E
 + Enter the RHS of the production:
 - Any more productions? (y/n) :- y
 + Enter the LHS of the production: F
 + Enter the RHS of the production: f
 - Any more productions? (y/n) :- y
 + Enter the LHS of the production: F
 + Enter the RHS of the production:
 - Any more productions? (y/n) :- n
 + Enter the starting non terminal: S
------------------FIRST-------------------
   B {'c'}
   C {'', 'b'}
   D {'', 'g', 'f'}
   E {'', 'g'}
   F {'', 'f'}
   S {'a'}
------------------FOLLOW------------------
   B {'b'}
   C {'b'}
   D {'h'}
   E {'f', 'h'}
   F {'h'}
   S {'$'}
```



```
PS D:\SPIT\SEM 6\CC Lab> py exe3.py
 + Enter the LHS of the production: S
 + Enter the RHS of the production: AaAb
 - Any more productions? (y/n) :- y
 + Enter the LHS of the production: S
 + Enter the RHS of the production: BbBa
 - Any more productions? (y/n) :- y
 + Enter the LHS of the production: A
 + Enter the RHS of the production:
 - Any more productions? (y/n) :- y
 + Enter the LHS of the production: B
 + Enter the RHS of the production:
 - Any more productions? (y/n) :- n
 + Enter the starting non terminal: S
------------------FIRST------------------
   A {''}
   B {''}
   S {'a', 'b'}
------------------FOLLOW-----------------
   A {'a', 'b'}
   B {'a', 'b'}
   S {'$'}
```



S->AaAb/BbBa

A-> ε

B-> ε

First(A)={ε}
First(B)={ε}
First(S)={First(A)- ε U First(a)} U{First(B)- ε U First(b)={a,b}

Follow(S) ={$}
Follow(A)={a,b}
Follow(B}={b,a}

- By Jaydeep Patil AISSMS's IOIT Pune

Conclusion:

From the above experiment, I was able to implement code and programmatically execute and verify the working of FIRST and FOLLOW SET by finding it for a given grammar.

Ref.:

- https://www.geeksforgeeks.org/first-set-in-syntax-analysis/
- https://www.geeksforgeeks.org/follow-set-in-syntax-analysis/