COMPUTING FIRST AND FOLLOW FOR THE GIVEN GRAMMAR

```
import sys
sys.setrecursionlimit(60)


def first(string):
    first_ = set()
    if string in non_terminals:
        alternatives = productions_dict[string]


        for alternative in alternatives:
            first_2 = first(alternative)
            first_ = first_ |first_2


    elif string in terminals:
        first_ = {string}


    elif string==" or string=='@':
        first_ = {'@'}


    else:
        first_2 = first(string[0])
        if '@' in first_2:
            i = 1
            while '@' in first_2:


                first_ = first_ | (first_2 - {'@'})
                if string[i:] in terminals:
```

```
                first_ = first_ | {string[i:]}
                break
            elif string[i:] == '':
                first_ = first_ | {'@'}
                break
            first_2 = first(string[i:])
            first_ = first_ | first_2 - {'@'}
            i += 1
        else:
            first_ = first_ | first_2

    return  first_


def follow(nT):
    follow_ = set()
    prods = productions_dict.items()
    if nT==starting_symbol:
        follow_ = follow_ | {'$'}
    for nt,rhs in prods:
        for alt in rhs:
            for char in alt:
                if char==nT:
                    following_str = alt[alt.index(char) + 1:]
                    if following_str=='':
                        if nt==nT:
                            continue
```

```
                else:
                    follow_ = follow_ | follow(nt)
            else:
                follow_2 = first(following_str)
                if '@' in follow_2:
                    follow_ = follow_ | follow_2-{'@'}
                    follow_ = follow_ | follow(nt)
                else:
                    follow_ = follow_ | follow_2
    return follow_




no_of_terminals=int(input("Enter no. of terminals: "))

terminals = []

print("Enter the terminals :")
for _ in range(no_of_terminals):
    terminals.append(input())

no_of_non_terminals=int(input("Enter no. of non terminals: "))

non_terminals = []
```

```
print("Enter the non terminals :")
for _ in range(no_of_non_terminals):
    non_terminals.append(input())

starting_symbol = input("Enter the starting symbol: ")

no_of_productions = int(input("Enter no of productions: "))

productions = []

print("Enter the productions:")
for _ in range(no_of_productions):
    productions.append(input())

productions_dict = {}

for nT in non_terminals:
    productions_dict[nT] = []

for production in productions:
    nonterm_to_prod = production.split("->")
    alternatives = nonterm_to_prod[1].split("/")
    for alternative in alternatives:
        productions_dict[nonterm_to_prod[0]].append(alternative)

FIRST = {}
FOLLOW = {}
```

```
for non_terminal in non_terminals:
    FIRST[non_terminal] = set()

for non_terminal in non_terminals:
    FOLLOW[non_terminal] = set()

for non_terminal in non_terminals:
    FIRST[non_terminal] = FIRST[non_terminal] | first(non_terminal)

FOLLOW[starting_symbol] = FOLLOW[starting_symbol] | {'$'}
for non_terminal in non_terminals:
    FOLLOW[non_terminal] = FOLLOW[non_terminal] | follow(non_terminal)

print("{: ^20}{: ^20}{: ^20}".format('Non Terminals','First','Follow'))
for non_terminal in non_terminals:
    print("{: ^20}{: ^20}{:
^20}".format(non_terminal,str(FIRST[non_terminal]),str(FOLLOW[non_terminal])))
```

OUTPUT

```
Enter the number of productions: 6
Enter Productions seperated by | and $ for epsilon
S -> ABCDE
A -> a | $
B -> b | $
C -> c
D -> d | $
E -> e | $
First of S:  ['a', 'b', 'c']
First of A:  ['a', '$']
First of B:  ['b', '$']
First of C:  ['c']
First of D:  ['d', '$']
First of E:  ['e', '$']

Follow of S :  ['$']
Follow of A :  ['b', 'c']
Follow of B :  ['c']
Follow of C :  ['d', 'e', '$']
Follow of D :  ['e', '$']
Follow of E :  ['$']
```