Experiment 5 - First and Follow

- Aim - To create a program that calculates the first and follow of the given grammar.
- Theory

→ First

It is a function that gives the set of terminals than begin the strings derived from the production rule

A symbol c is in first(α) if and only if:

$$\alpha \rightarrow c\beta$$

for some sequence β of grammar symbols.

→ Follow

Follow (A) is defined as the collection of terminal symbols that occur directly to the right of A.

Follow (A) = {a | S ⇒ αAaβ where α, β can be any strings}

- Algorithm

→ Calculation of first

- Declare and define a function first with grammar and the given non-terminal as two parameters

- Create an empty list f to store the first values.
- Check if the given left hand side non terminal is not present in grammar, if true return lhs
- For each i in gram[lhs], repeat the following steps:
- Check if i[0] is not in gram, if to check if it is a terminal value, if true append it to list f
- Else if check if i[0] is in gram, if true call the function first(gram, i[0]) recursively and add the value to first of f
- End of loop
- Return the list f
- End of function first.

⇒ Calculation of follow:

- Declare and define a function follow with gram, the non terminal symbol on lhs - term and start symbol as parameters

- Create an empty list a to store values of follow
- For a given rule [lhs non-terminal] in gram, repeat the following steps:

→ Check if rule is the start element, if true add $ to a

→ For i in gram [rule], repeat the following steps:

- Check if right hand side i has the given non-terminal value term, if true set temp to i
- Calculate the index of the term in i and store in indx
- Check if indx + 1 is not equal to length of list i, if true check if i[-1] is in firsts, then add first [i[-1]] to list a
- Else add i[-1] to a, and add e to a
- Check if rule != term and e is in a, if true add follow (gram, rule, start) to a
- End of nested loops.
- Return the list a and ~~retu~~ end of function follow.

Program

```
def add_dict(d,k,l):
    d[k]=list()
    d[k].extend(l)
    return d

prod = {}
n=int(input("Enter the number of non-terminals: "))
key = input("Enter the non-terminal: ")
start=key
for i in range (0,n):
    rhs=[]
    x=int(input("Enter no of terms on rhs: "))
    for j in range (0,x):
        value = input("Enter RHS term: ")
        rhs.append(value)
    prod=add_dict(prod,key,rhs)
    if(i!=n-1):
        key = input("Enter the non-terminal:")


result = prod
print("\nDictionary of Production rules:",result,"\n\n")

terminals= []
for i in result:
    for j in result[i]:
        for k in j:
            if k not in result:
                terminals+=[k]
```

```
terminals = list(set(terminals))
print("List of terminals:",terminals)
print("\n")

def first(gram, lhs):
    f = []
    if lhs not in gram:
        return [lhs]
    for i in gram[lhs]:
        if i[0] not in gram:
            f.append(i[0])
        elif i[0] in gram:
            f += first(gram, i[0])
    return f

firsts = {}
for i in result:
    firsts[i] = first(result,i)
    print(f'First({i}):',firsts[i])

def follow(gram, term,start):
    a = []
    for rule in gram:
        if rule == start:
            a+=['$']
        for i in gram[rule]:
            if term in i:
                temp = i
                indx = i.index(term)
                if indx+1!=len(i):
```

```
                if i[-1] in firsts:
                    a+=firsts[i[-1]]
                else:
                    a+=[i[-1]]
            else:
                a+=["e"]
            if rule != term and "e" in a:
                a+= follow(gram,rule,start)

    return a
print('\n\n')
follows = {}
x=0
for i in result:
    follows[i] = list(set(follow(result,i,start)))
    if "e" in follows[i]:
        follows[i].pop(follows[i].index("e"))
    print(f'Follow({i}):',follows[i])
```

Output

```
Enter the number of non-terminals: 5
Enter the non-terminal: E
Enter no of terms on rhs: 1
Enter RHS term: TX
Enter the non-terminal:X
Enter no of terms on rhs: 2
Enter RHS term: +TX
Enter RHS term: e
Enter the non-terminal:T
Enter no of terms on rhs: 1
Enter RHS term: FY
Enter the non-terminal:Y
Enter no of terms on rhs: 2
Enter RHS term: *FY
Enter RHS term: e
Enter the non-terminal:F
Enter no of terms on rhs: 2
Enter RHS term: (E)
Enter RHS term: i

Dictionary of Production rules: {'E': ['TX'], 'X': ['+TX', 'e'], 'T': ['FY'], 'Y': ['*FY', 'e'], 'F': ['(E)', 'i']}


List of terminals: ['e', 'i', '(', '*', '+', ')']


First(E): ['(', 'i']
First(X): ['+', 'e']
First(T): ['(', 'i']
First(Y): ['*', 'e']
First(F): ['(', 'i']



Follow(E): ['$', ')']
Follow(X): ['$', ')']
Follow(T): ['$', '+', ')']
Follow(Y): ['$', '+', ')']
Follow(F): ['$', '*', '+', ')']
```

- Result

A program to calculate first and follow was implemented in python