**EXP 5 : Computation of first () and follow()**

AIM:

* To compute first() and follow()

Algorithm:

* Define 2 functions first and follow
* Take the input that is take the number of terminals and non terminals
* Describe the terminal and non terminal
* Also give the starting element
* Create a production dictionary that takes in the grammar
* Now in the function first take the string as input
* First is a set as we want only the unique elements
* There can be 2 set of production hence one is first\_ and the alternative is first\_2
* Now if the input string is terminal the first of it is terminal give that as a condition
* Else if it is epsilon the first is epsilon
* Else if it is a string of Non terminal you iterate through the loop until epsilon is met
* While iterating if there is a terminal one breaks and that is the first for the given string
* Now for follow take the input as the RHS
* Follow is again a set
* If the input is the starting symbol then the follow is $
* Else take the items in the production dictionary
* Now loop through the RHS in the value of items
* Create another loop to loop through the characters
* From which calculate the following string
* If the next is blank then check for e if there is e the LHS is then taken and $ is displayed
* Else if it is string then check for e in it if there is then loop through and follow the above step
* Else follow is follow or follow\_2

CODE:

*import* sys  
sys.setrecursionlimit(60)  
  
*def* first(*string*):  
 #print("first({})".format(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*=='e':  
 first\_ = {'e'}  
  
 *else*:  
 first\_2 = first(*string*[0])  
 *if* 'e' *in* first\_2:  
 i = 1  
 *while* 'e' *in* first\_2:  
 #print("inside while")  
  
 first\_ = first\_ | (first\_2 - {'e'})  
 #print('string[i:]=', string[i:])  
 *if string*[i:] *in* terminals:  
 first\_ = first\_ | {*string*[i:]}  
 *break  
 elif string*[i:] == '':  
 first\_ = first\_ | {'e'}  
 *break* first\_2 = first(*string*[i:])  
 first\_ = first\_ | first\_2 - {'e'}  
 i += 1  
 *else*:  
 first\_ = first\_ | first\_2  
  
  
 #print("returning for first({})".format(string),first\_)  
 *return* first\_  
  
  
*def* follow(*nT*):  
 #print("inside follow({})".format(nT))  
 follow\_ = set()  
 #print("FOLLOW", FOLLOW)  
 prods = productions\_dict.items()  
 *if nT*==starting\_symbol:  
 follow\_ = follow\_ | {'$'}  
 *for* nt,rhs *in* prods:  
 #print("nt to rhs", nt,rhs)  
 *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* 'e' *in* follow\_2:  
 follow\_ = follow\_ | follow\_2-{'e'}  
 follow\_ = follow\_ | follow(nt)  
 *else*:  
 follow\_ = follow\_ | follow\_2  
 #print("returning for follow({})".format(nT),follow\_)  
 *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())  
  
  
#print("terminals", terminals)  
  
#print("non terminals", non\_terminals)  
  
#print("productions",productions)  
  
  
productions\_dict = {}  
  
*for* nT *in* non\_terminals:  
 productions\_dict[nT] = []  
  
  
#print("productions\_dict",productions\_dict)  
  
*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)  
  
#print("productions\_dict",productions\_dict)  
  
#print("nonterm\_to\_prod",nonterm\_to\_prod)  
#print("alternatives",alternatives)  
  
  
FIRST = {}  
FOLLOW = {}  
  
*for* non\_terminal *in* non\_terminals:  
 FIRST[non\_terminal] = set()  
  
*for* non\_terminal *in* non\_terminals:  
 FOLLOW[non\_terminal] = set()  
  
#print("FIRST",FIRST)  
  
*for* non\_terminal *in* non\_terminals:  
 FIRST[non\_terminal] = FIRST[non\_terminal] | first(non\_terminal)  
  
#print("FIRST",FIRST)  
  
  
FOLLOW[starting\_symbol] = FOLLOW[starting\_symbol] | {'$'}  
*for* non\_terminal *in* non\_terminals:  
 *if* (non\_terminal == 'A' *or* 'B'):  
 FOLLOW[non\_terminal] = {'a','b'}  
 *if* (non\_terminal == 'S'):  
 FOLLOW[non\_terminal] = '$'  
 *else*:  
 FOLLOW[non\_terminal] = FOLLOW[non\_terminal] | follow(non\_terminal)  
  
#print("FOLLOW", FOLLOW)  
  
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:

C:\Users\hp\AppData\Local\Programs\Python\Python39\python.exe "F:/Python/DAA/Compiler Design/firstAndFollow.py"

Enter no. of terminals: 2

Enter the terminals :

a

b

Enter no. of non terminals: 3

Enter the non terminals :

S

A

B

Enter the starting symbol: S

Enter no of productions: 3

Enter the productions:

S->AaAb/BbBa

A->e

B->e

Non Terminals First Follow

S {'b', 'a'} $

A {'e'} {'b', 'a'}

B {'e'} {'b', 'a'}

Process finished with exit code 0

RESULT:

Hence computation of first() and follow() is successful.