**EXPERIMENT 07**

CLASS: TE CMPN A ROLL NO. : 19

NAME: REBECCA DIAS PID: 182027

**Aim:** To find the first and follow sets of a given grammar.

**Theory:**

What are first and follow sets?

The construction of a predictive parser is aided by two functions associated with a grammar G. These functions, FIRST and FOLLOW, allow us to fill in the entries of a predictive parsing table for G.

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) }

* 1. If FIRST (Yi) contains Є for all i = 1 to n, then add Є to FIRST(X).

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).

1. If A->pB is a production, then everything in FOLLOW(A) is in FOLLOW(B).
2. If A->pBq is a production and FIRST(q) contains Є,

then FOLLOW(B) contains { FIRST(q) – Є } U FOLLOW(A)

How are these sets used by the parser?

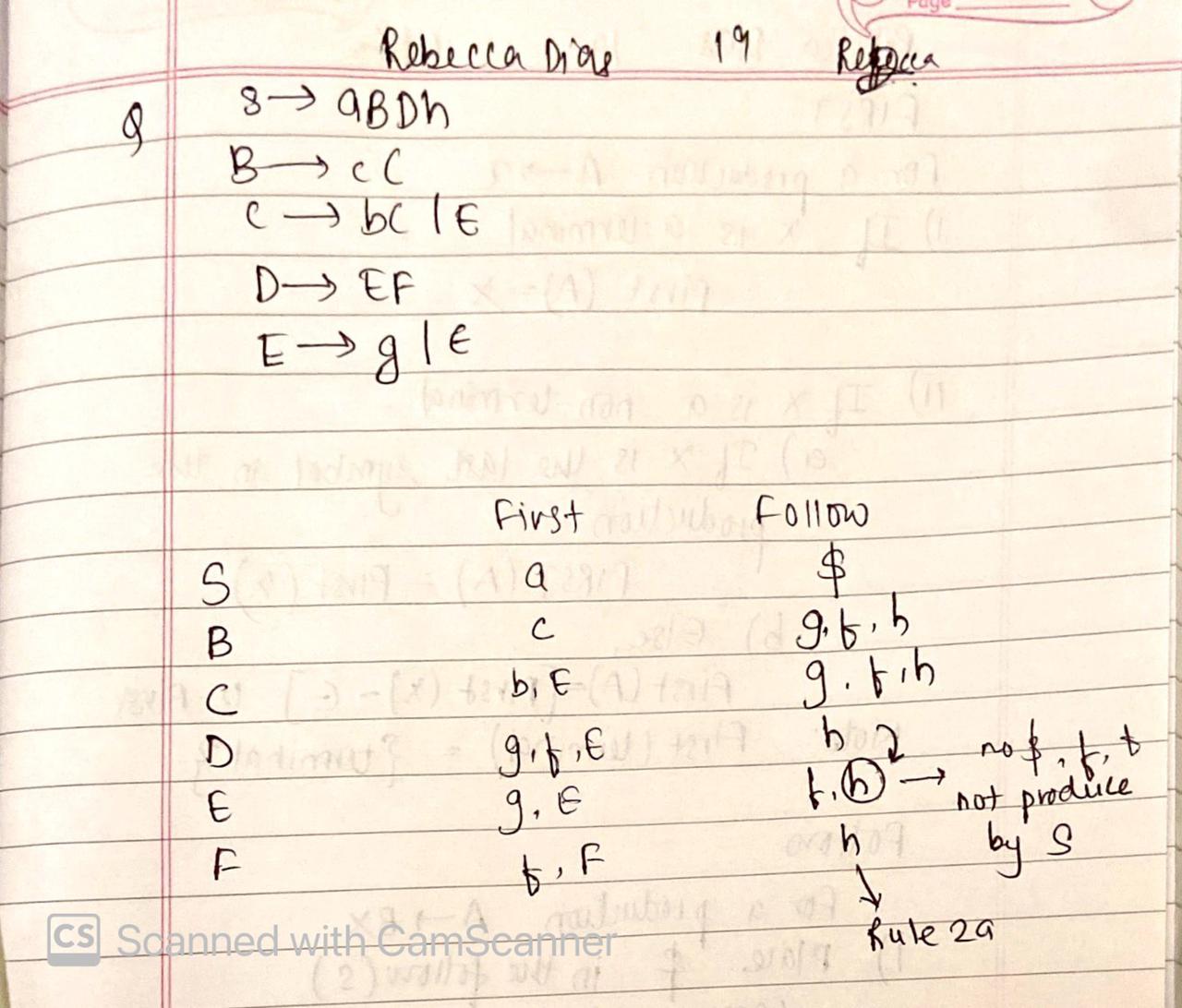
To check if given grammar is LL(1) or not, we need to construct a parse table. This table can be constructed using FIRST and FOLLOW sets. Given below are the rules on how to construct the parse table using FIRST and FOLLOW sets – For each production X → α

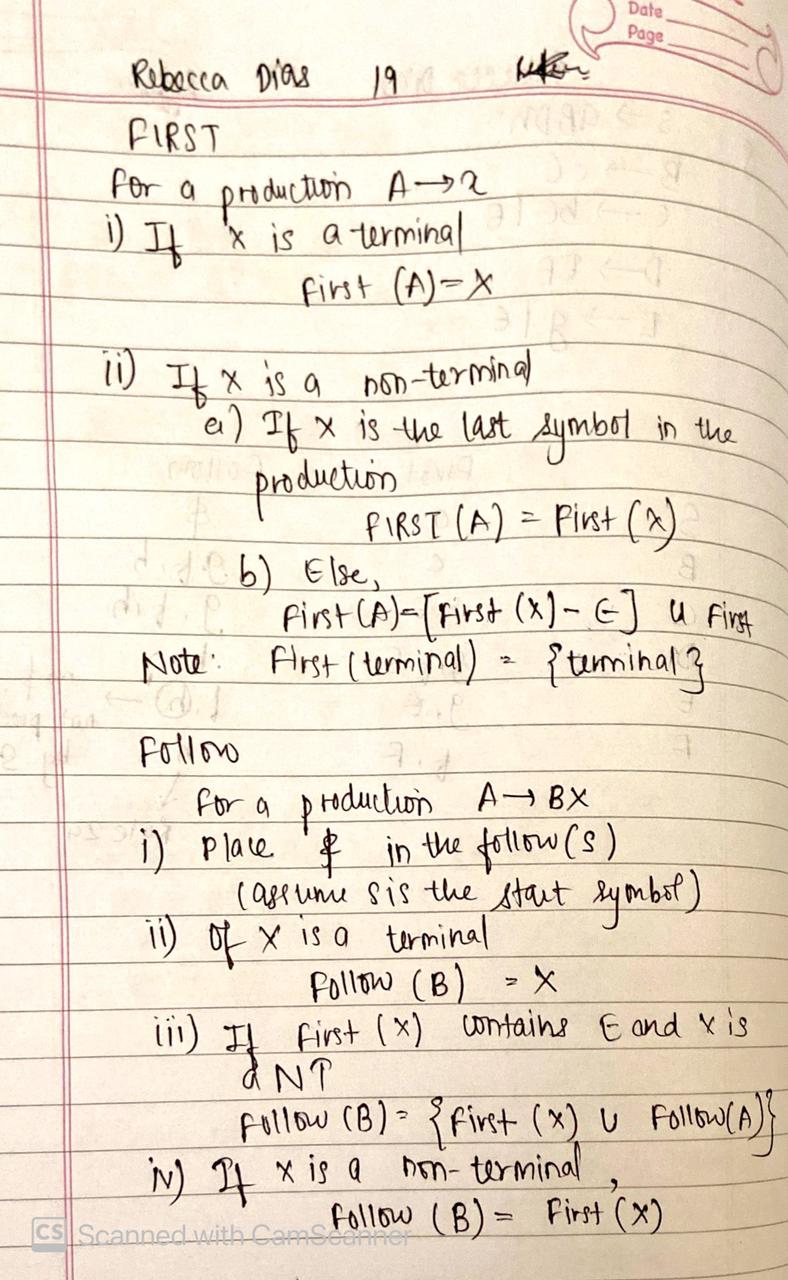
* for each terminal t in First(α): put α in table[X,t]
* if epsilon is in First(α) then:

for each terminal t in Follow(X): put α in table[X,t]

The grammar is not LL(1) if and only if there is more than one entry for any cell in the table.

Find the FIRST and FOLLOW of a grammar –





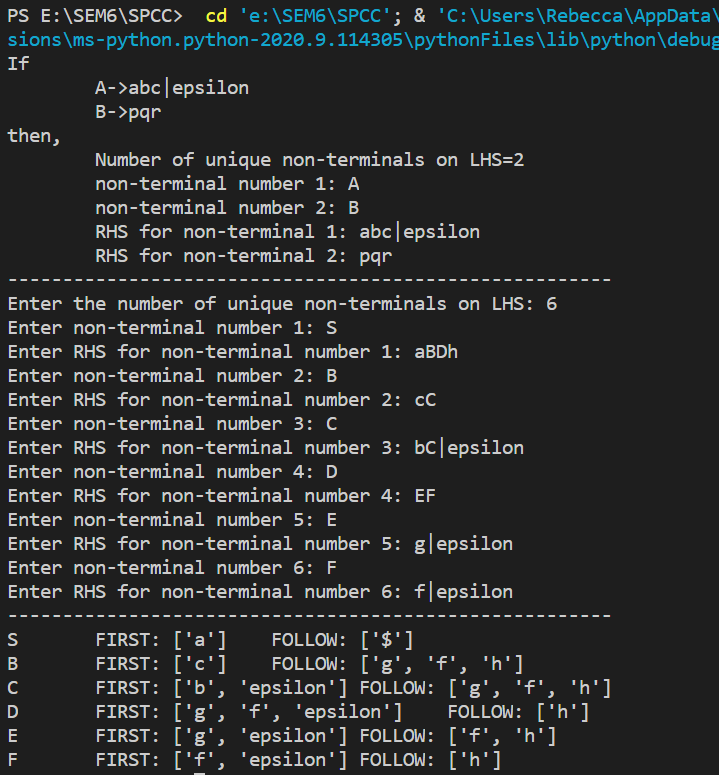
**Implementation:**

Psuedo code

# CODE:

|  |
| --- |
| print("If\n\tA->abc|epsilon\n\tB->pqr\nthen,\n\tNumber of unique non-terminals on LHS=2\n\tnon-terminal number 1: A\n\tnon-terminal number 2: B\n\tRHS for non-terminal 1: abc|epsilon\n\tRHS for non-terminal 2: pqr")  print("-"\*55)  num=int(input("Enter the number of unique non-terminals on LHS: "))  nt\_list=[]  production\_list=[]  first\_list=[]  follow\_list=[]  for i in range(num):      nt=input(f"Enter non-terminal number {i+1}: ")      production=input(f"Enter RHS for non-terminal number {i+1}: ")      nt\_list.append(nt)      production\_list.append(production)      first\_list.append([])      follow\_list.append([])  follow\_list[0].append('$')  def nt\_present(f\_list,production,index,j):      for first\_value in first\_list[index]:          if (first\_value=="epsilon"):              if (j==len(production)-1):                  f\_list.append("epsilon")              elif (production[j+1] in nt\_list):                  nt\_present(f\_list,production,nt\_list.index(production[j+1]),j+1)              elif (production[j+1] not in nt\_list):                  f\_list.append(production[j+1])          else:              f\_list.append(first\_value)      return    for i in reversed(range(num)):      nt=nt\_list[i]      productions=production\_list[i]      t\_productions=productions.split("|")      j=0      for production in t\_productions:          if (production=="epsilon"):              first\_list[i].append(production)              continue          elif (production[j] not in nt\_list):              first\_list[i].append(production[j])          else:              index=nt\_list.index(production[j])              nt\_present(first\_list[i],production,index,j)    def follow\_nt\_present(p\_idx,f\_list,production,j):      nt\_index=nt\_list.index(production[j])      for first in first\_list[nt\_index]:          if (first=="epsilon"):              if (j==len(production)-1):                  for follow in follow\_list[p\_idx]:                      f\_list.append(follow)              elif (production[j+1] not in nt\_list):                  f\_list.append(production[j+1])              else:                  follow\_nt\_present(p\_idx,f\_list,production,j+1)          else:              f\_list.append(first)      return  for prod\_idx,productions in enumerate(production\_list):      for production in productions.split("|"):              if (production=="epsilon"):                  continue              else:                  for j in range(len(production)):                      if (production[j] in nt\_list):                          nt\_index=nt\_list.index(production[j])                          if (j==len(production)-1):                              if (nt\_index==prod\_idx):                                  continue                              for follow\_value in follow\_list[prod\_idx]:                                  follow\_list[nt\_index].append(follow\_value)                          elif (production[j+1] not in nt\_list):                              follow\_list[nt\_index].append(production[j+1])                          else:                              follow\_nt\_present(prod\_idx,follow\_list[nt\_index],production,j+1)  print("-"\*55)  for i in range(num):      print(f"{nt\_list[i]}\tFIRST: {first\_list[i]}\tFOLLOW: {follow\_list[i]}") |

# OUTPUT:



**CONCLUSION:**

The first and follow sets can be found for any given grammar.