**Code:**

from tabulate import tabulate  
  
  
separator = '->'  
epsilon = 'e'  
  
nt = int(input("Enter the No. of Non terminals: "))  
t = int(input("Enter the No. of terminals: "))  
# productions = []  
# terminals = []  
# variables = []  
  
variables = []  
terminals = []  
for i in range(0, nt):  
 variables.append(input("Enter the Non Terminal Name/Symbol: "))  
for i in range(0, t):  
 terminals.append(input("Enter the Terminal Name/Symbol: "))  
p = int(input("Enter the No. of productions: "))  
productions = []  
  
for i in range(0,p):  
 productions.append(input())  
  
FIRST = {}  
for i in range(0,nt):  
 print("Enter the FIRST of ", variables[i])  
 FIRST[variables[i]] = input().split(",")  
  
FOLLOW = {}  
  
for i in range(0,nt):  
 print("Enter the FOLLOW of ", variables[i])  
 FIRST[variables[i]] = input().split(",")  
  
print(FIRST[variables[0]])  
# countTerminals = int(input("Enter no. of terminals : "))  
# print("Enter terminals : ")  
# for i in range(0, countTerminals):  
# terminals.append(input())  
# terminals.append('$')  
  
# for i in range(no\_of\_prod):  
 # temp = input(f"Production {i+1} : ").replace(" ", "").split(separator)  
  
parseTable = []  
temp = [''] + terminals  
parseTable.append(temp)  
  
for j in FIRST.keys():  
 row = ['' for x in terminals]  
 # print(row)  
 row = [j] + row  
 # print(row)  
 for i in FIRST[j]:  
 for k in terminals:  
 # print(k, ' ', i)  
 if (i == epsilon):  
 # print('null', terminals.index(k))  
 for z in FOLLOW[j]:  
 if z == k:  
 row[terminals.index(k) + 1] = f'{j} -> {epsilon}'  
 # print(terminals.index(k) + 1)  
 # print(row[terminals.index(k) + 1])  
 elif(i == k):  
 # print('match', terminals.index(k))  
 production = ''  
 for prod in productions:  
 if prod.startswith(j):  
 production = prod  
 break  
 temp = production.split(separator)[1].split('|')  
 # print(temp)  
 production = temp[0]  
 for x in temp:  
 if x.startswith(i):  
 production = x  
 # print("I am using FISRT")  
 # print(terminals.index(k) + 1)  
 row[terminals.index(k) + 1] = f'{j} -> {production}'  
 parseTable.append(row)  
  
print(tabulate(parseTable, tablefmt="simple\_grid"))

def validateStringUsingStackBuffer(parsing\_table, grammarll1,table\_term\_list, input\_string,term\_userdef,start\_symbol):  
 print(f"\nValidate String => {input\_string}\n")  
 if grammarll1 == False:  
 return f"\nInput String = " \  
 f"\"{input\_string}\"\n" \  
 f"Grammar is not LL(1)"  
 stack = [start\_symbol, '$']  
 buffer = []  
 input\_string = input\_string.split()  
 input\_string.reverse()  
 buffer = ['$'] + input\_string  
 print("{:>20} {:>20} {:>20}".  
 format("Buffer", "Stack","Action"))  
 while True:  
 if stack == ['$'] and buffer == ['$']:  
 print("{:>20} {:>20} {:>20}".format(' '.join(buffer),' '.join(stack),"Valid"))  
 return "\nValid String!"  
 elif stack[0] not in term\_userdef:  
 x = list(diction.keys()).index(stack[0])  
 y = table\_term\_list.index(buffer[-1])  
 if parsing\_table[x][y] != '':  
 entry = parsing\_table[x][y]  
 print("{:>20} {:>20} {:>25}".format(' '.join(buffer),' '.join(stack),f"T[{stack[0]}][{buffer[-1]}] = {entry}"))  
 lhs\_rhs = entry.split("->")  
 lhs\_rhs[1] = lhs\_rhs[1].replace('#', '').strip()  
 entryrhs = lhs\_rhs[1].split()  
 stack = entryrhs + stack[1:]  
 else:  
 return f"\nInvalid String! No rule at " \  
 f"Table[{stack[0]}][{buffer[-1]}]."  
 else:  
 if stack[0] == buffer[-1]:  
 print("{:>20} {:>20} {:>20}".format(' '.join(buffer),' '.join(stack),f"Matched:{stack[0]}"))  
 buffer = buffer[:-1]  
 stack = stack[1:]  
 else:  
 return "\nInvalid String! " \  
 "Unmatched terminal symbols"

Output: 





