Shri Ramdeobaba College of Engineering and Management, Nagpur
Department of Computer Science and Engineering
Session: 2021-2022 [EVEN SEM]

Compiler Design Lab **Name:- Krishna Mundada** **E-45**

PRACTICAL No. 3

Topic: Parser Construction

Platform: Windows or Linux

Language to be used: Python or Java (based on the companies targeted for placement)

Aim: To find FIRST and Follow of a grammar.

(A) Write a program to find FIRST for any grammar. All the following rules of FIRST must be implemented.

For a generalized grammar: A   αXY

FIRST (A) = FIRST (αXY)

= α if α is the terminal symbol (Rule-1)

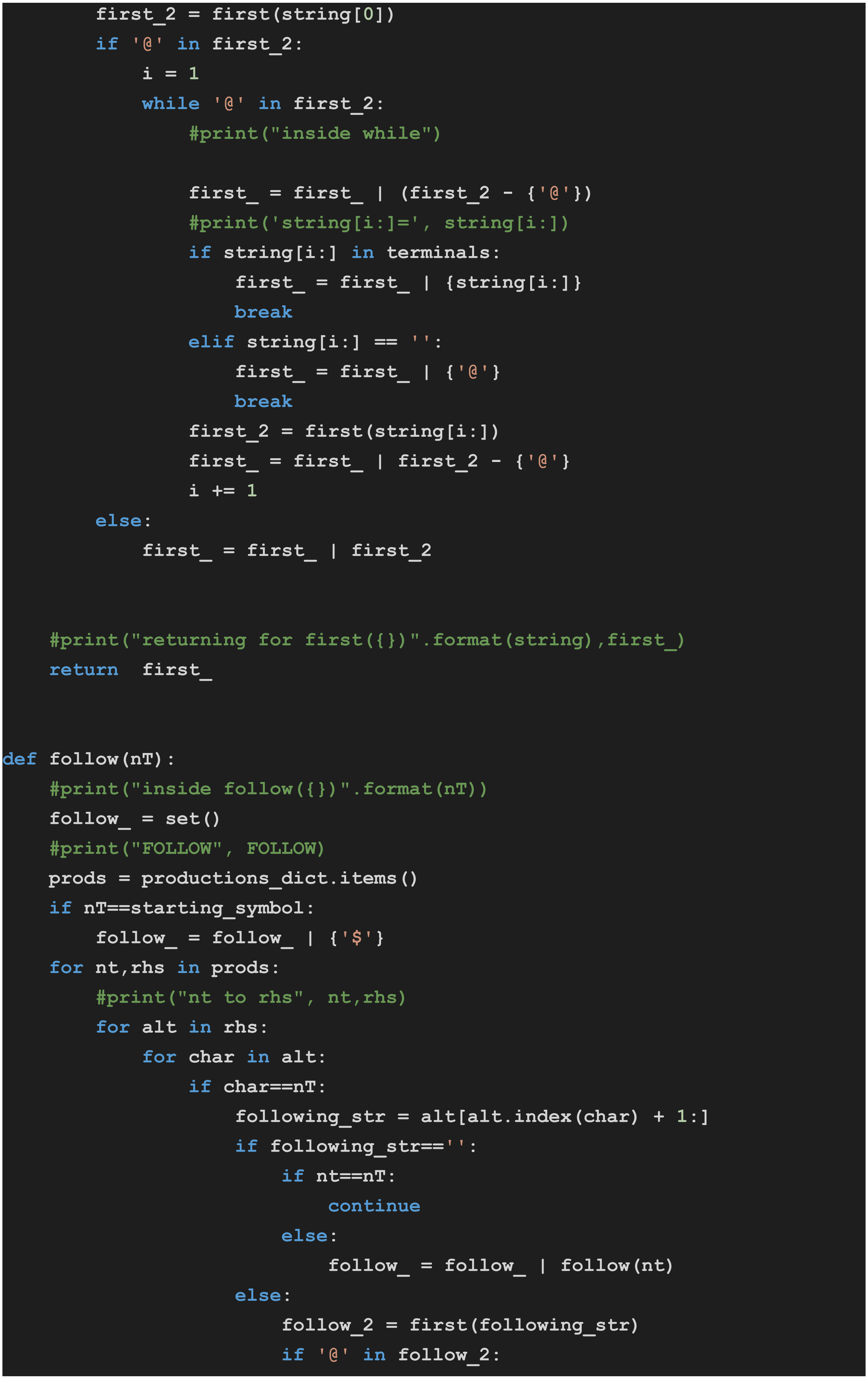
= FIRST (α) if α is a non-terminal and FIRST (α) does not contain ε (Rule-2)

= FIRST (α) - ε ∪ FIRST (XY) if a is a non-terminal and FIRST (α) contains ε (Rule-3)

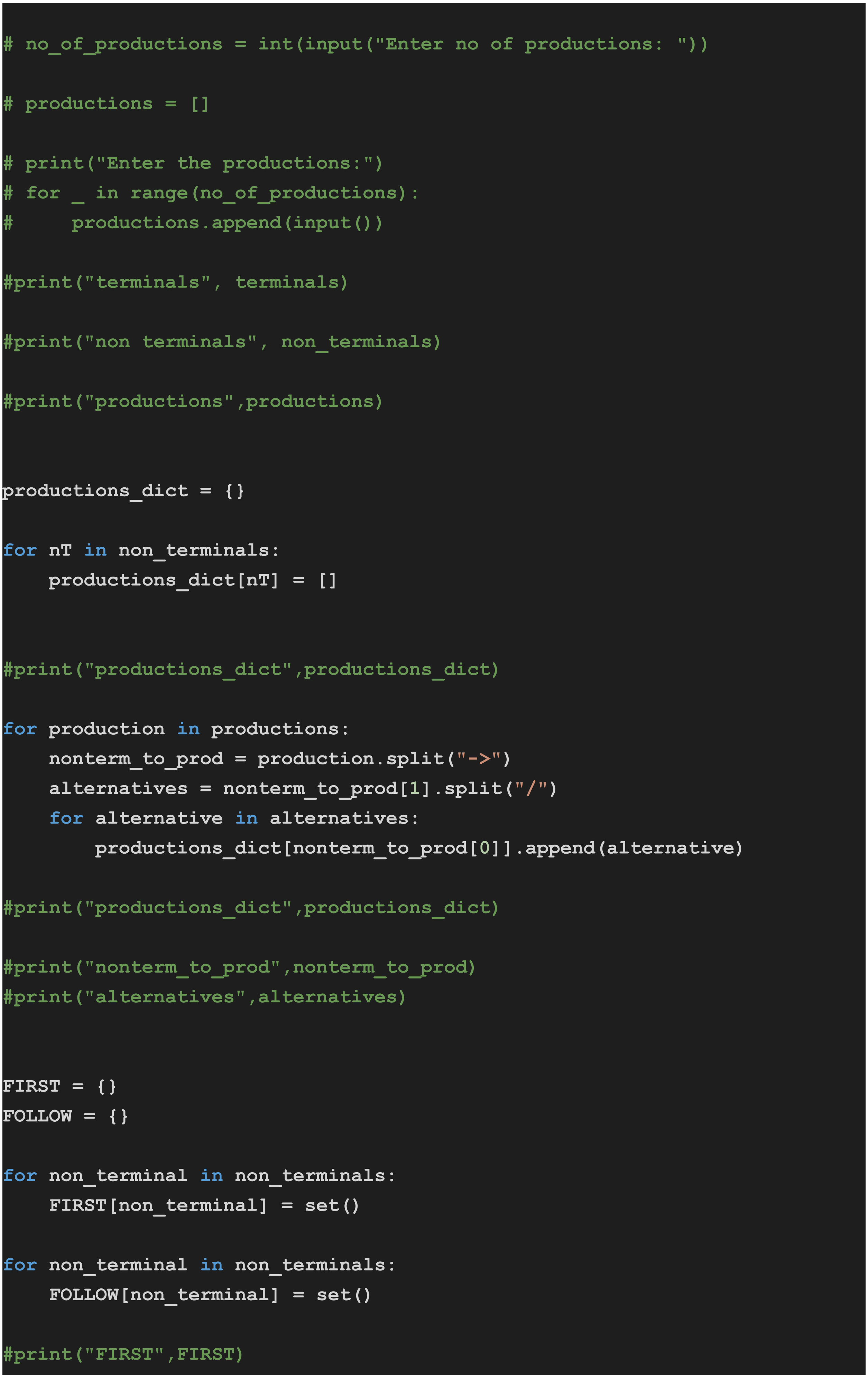
Input: Grammar rules from a file or from console entered by user.

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=='@':
        first_ = {'@'}

    else:
```



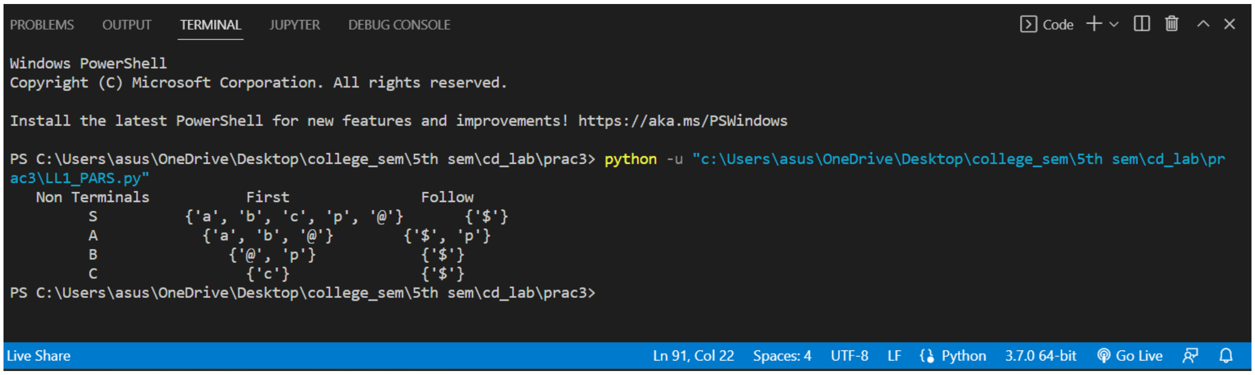
```
        first_2 = first(string[0])
        if '@' in first_2:
            i = 1
            while '@' in first_2:
                #print("inside while")

                first_ = first_ | (first_2 - {'@'})
                #print('string[i:]=', string[i:])
                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


    #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 '@' in follow_2:
                            follow_ = follow_ | follow_2-{'@'}
                            follow_ = follow_ | follow(nt)
                        else:
                            follow_ = follow_ | follow_2
    #print("returning for follow({})".format(nT),follow_)
    return follow_





no_of_terminals=4

terminals = ['a','b','c','p']

no_of_non_terminals=4

non_terminals = ['S','A','B','C']

starting_symbol = 'S'

no_of_productions = 4

productions = ['S->AB/C',
            'A->a/BC',
            'B->p/@',
             'C->c'
            ]

# 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:
    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
])))
```

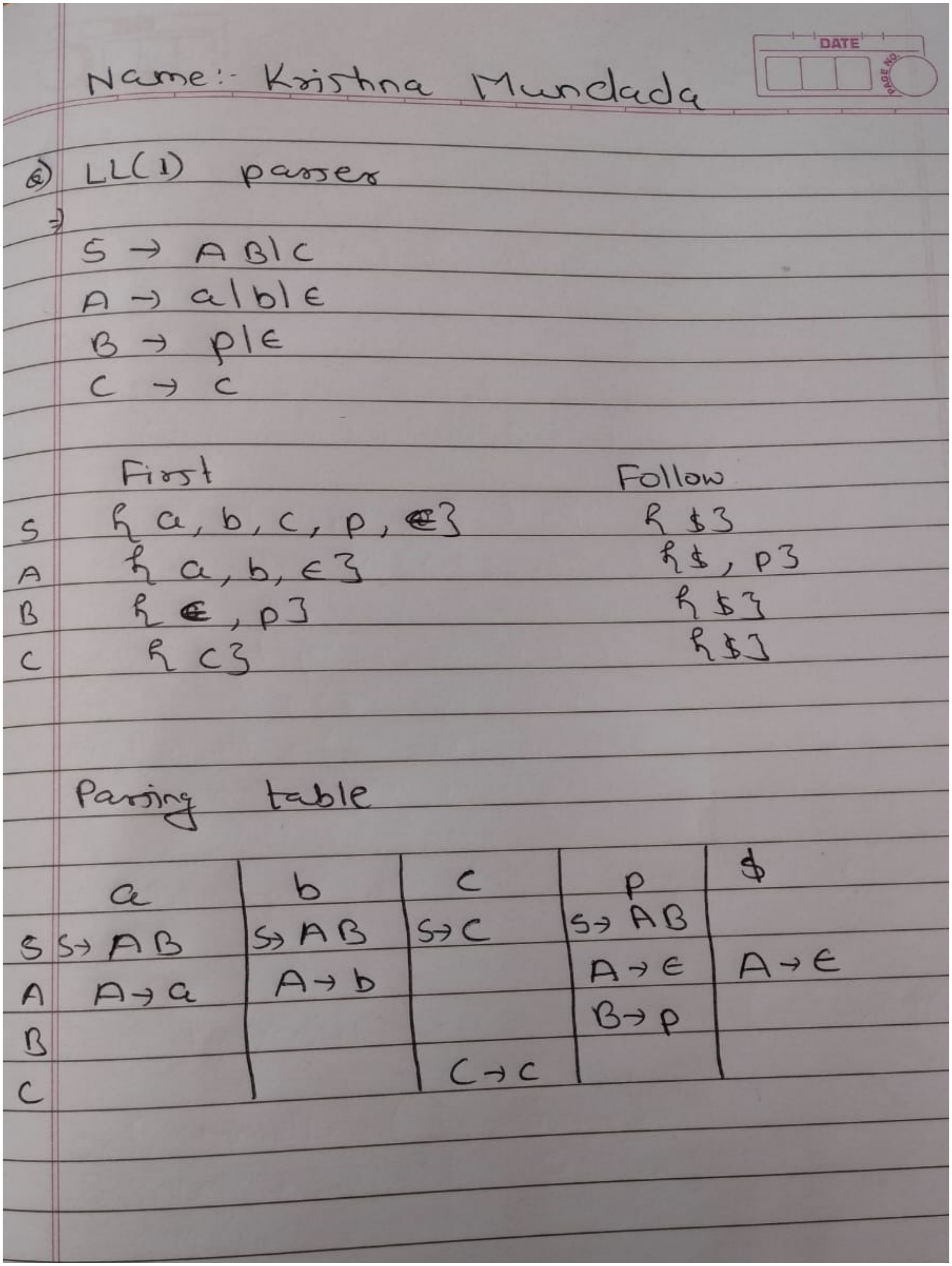
Implementation: FIRST rules
Output: FIRST information for each non-terminal



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\asus\OneDrive\Desktop\college_sem\5th sem\cd_lab\prac3> python -u "c:\Users\asus\OneDrive\Desktop\college_sem\5th sem\cd_lab\pr
ac3\LL1_PARS.py"
   Non Terminals           First               Follow
         S          {'a', 'b', 'c', 'p', '@'}       {'$'}
         A            {'a', 'b', '@'}        {'$', 'p'}
         B               {'@', 'p'}            {'$'}
         C                 {'c'}               {'$'}
PS C:\Users\asus\OneDrive\Desktop\college_sem\5th sem\cd_lab\prac3>
```

(B) Calculate Follow for the given grammar manually, input the follow information and Construct the LL (1) parsing table using the FIRST and FOLLOW values computed above.

⇒



Name:- Krishna Mundada

Q) LL(1) parser

⇒

S → AB|C

A → a|b|ε

B → p|ε

C → c

	First	Follow
S	{a, b, c, p, ε}	{$}
A	{a, b, ε}	{$, p}
B	{ε, p}	{$}
C	{c}	{$}

Parsing table

	a	b	c	p	$
S	S→AB	S→AB	S→C	S→AB	
A	A→a	A→b		A→ε	A→ε
B				B→p	
C			C→c		