**Compiler Design(18CSC304J)**

**Experiment 7**

**Computation of First**

Harsh Goel

RA1811003010185

**Aim:** To study and implement Computation of First.

**Language: Python**

**Procedure:**

1. Create a file or select the file for performing the operations on.
2. Start any python IDE and type the necessary code.
3. Run the code and perform the operations required.
4. Note the output and document it.

**Algorithm:**

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,
4. FIRST(X) = FIRST(Y1)
5. If FIRST(Y1) contains Є then FIRST(X) = { FIRST(Y1) – Є } U { FIRST(Y2) }
6. If FIRST (Yi) contains Є for all i = 1 to n, then add Є to FIRST(X).

**Code Snippet:**

import sys

sys.setrecursionlimit(60)

def first(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

    return first\_

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())

productions\_dict = {}

for nT in non\_terminals:

    productions\_dict[nT] = []

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)

FIRST = {}

for non\_terminal in non\_terminals:

    FIRST[non\_terminal] = set()

for non\_terminal in non\_terminals:

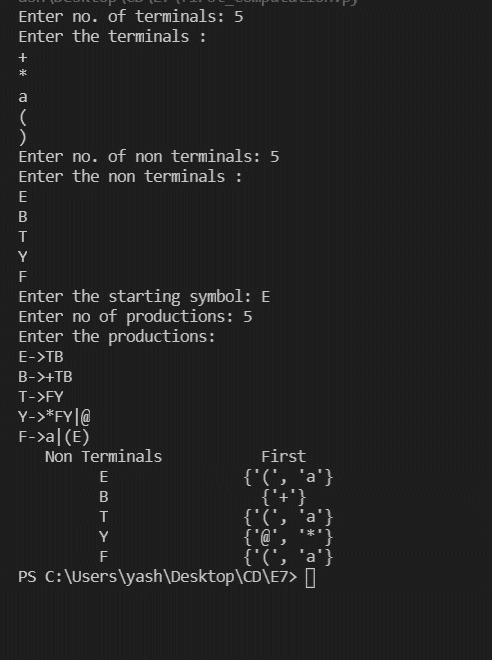
    FIRST[non\_terminal] = FIRST[non\_terminal] | first(non\_terminal)

print("{: ^20}{: ^20}".format('Non Terminals', 'First'))

for non\_terminal in non\_terminals:

    print("{: ^20}{: ^20}".format(non\_terminal, str(FIRST[non\_terminal]), ))

**Output Screenshots:**

****

**Result:**

The code was successfully implemented in Python and output was recorded.