Experiment 6 : Predictive Parsing Table

• Aim - To create a program that calculates the predictive parsing table for the given grammar.

• Theory

Rules for construction of Predictive Parsing Table:

→ $A \rightarrow \alpha$
First $(\alpha) = \{a_1, a_2, a_3\}$
Copy the rule to $[A, a_1]$ $[A, a_2]$ $[A, a_3]$

→ $A \rightarrow \epsilon$
Follow $(A) = \{a_1, a_2, a_3\}$
Copy the rule to $[A, a_1]$ $[A, a_2]$ $[A, a_3]$

• Algorithm

→ Store the list of terminals in a variable term and pop e from the list, then add $

→ Create an empty dictionary table.

→ For i in result (which stores the whole grammar) repeat the following steps:

→ For j in term repeat the following:

→ Check if j is in first [i], if true set table [i] [j] to ~~se~~ result [i(0) (0)], else set table [i] [j] to empty string.

Date
Page

→ Check if e in first (i) then for every j in term, check if j is in follow [i], if true add e to table [i] [j]

→ End of nested loops.

→ Print the list of terminals horizontally.

→ For i in result and for every j in term, check if table (i) [j] is not blank if true print the table value

→ Else print empty string.

- Result

→ A program to compute the predictive parsing table was implemented in python

Scanned with CamScanner

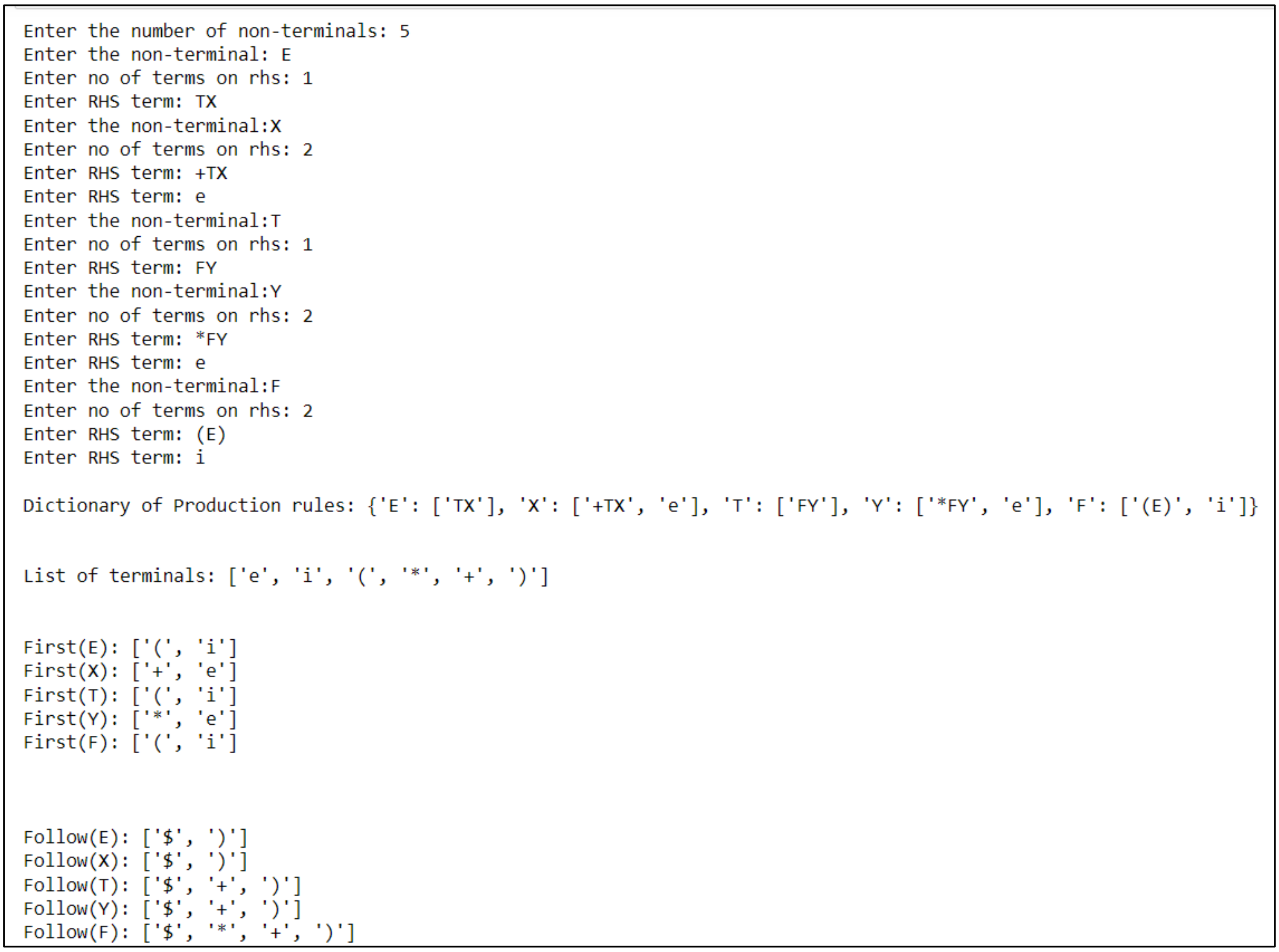
Program

```
def add_dict(d,k,l):
    d[k]=list()
    d[k].extend(l)
    return d

prod = {}
n=int(input("Enter the number of non-terminals: "))
key = input("Enter the non-terminal: ")
start=key
for i in range (0,n):
    rhs=[]
    x=int(input("Enter no of terms on rhs: "))
    for j in range (0,x):
        value = input("Enter RHS term: ")
        rhs.append(value)
    prod=add_dict(prod,key,rhs)
    if(i!=n-1):
        key = input("Enter the non-terminal:")



result = prod
print("\nDictionary of Production rules:",result,"\n\n")

terminals= []
for i in result:
    for j in result[i]:
        for k in j:
            if k not in result:
                terminals+=[k]
```

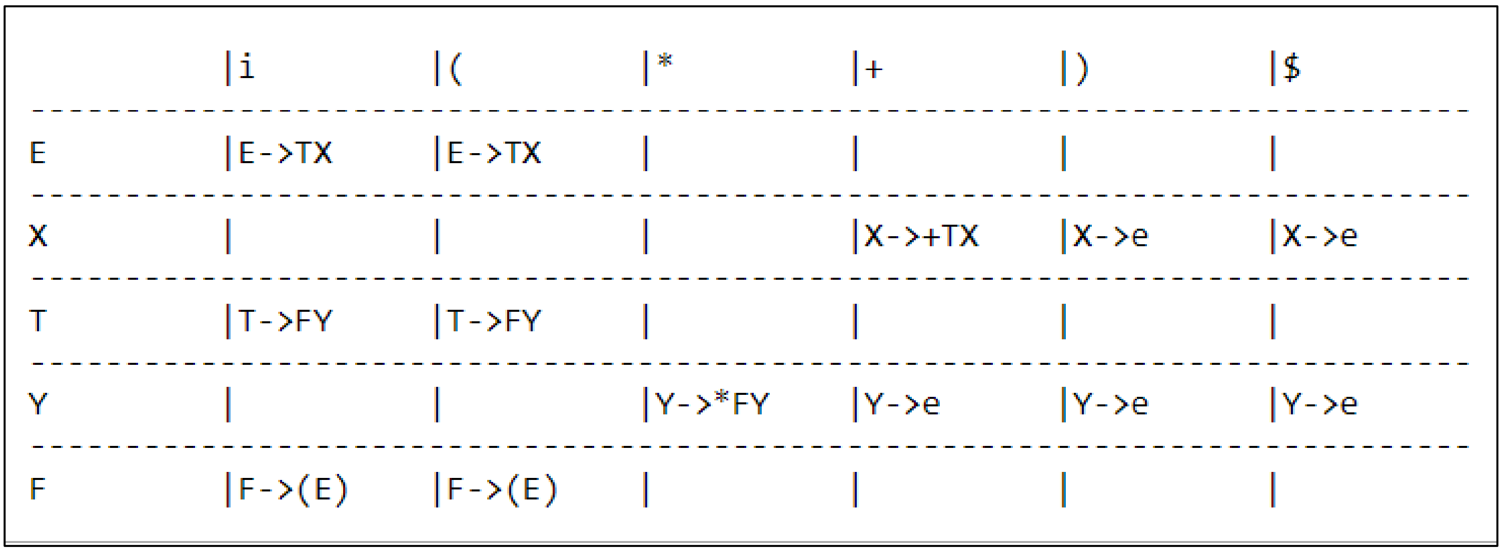
```
terminals = list(set(terminals))
print("List of terminals:",terminals)
print("\n")

def first(gram, lhs):
    f = []
    if lhs not in gram:
        return [lhs]
    for i in gram[lhs]:
        if i[0] not in gram:
            f.append(i[0])
        elif i[0] in gram:
            f += first(gram, i[0])
    return f

firsts = {}
for i in result:
    firsts[i] = first(result,i)
    print(f'First({i}):',firsts[i])

def follow(gram, term,start):
    a = []
    for rule in gram:
        if rule == start:
            a+=['$']
        for i in gram[rule]:
            if term in i:
                temp = i
                indx = i.index(term)
                if indx+1!=len(i):
```

```
                if i[-1] in firsts:
                    a+=firsts[i[-1]]
                else:
                    a+=[i[-1]]
            else:
                a+=["e"]
            if rule != term and "e" in a:
                a+= follow(gram,rule,start)

    return a
print('\n\n')
follows = {}
x=0
for i in result:
    follows[i] = list(set(follow(result,i,start)))
    if "e" in follows[i]:
        follows[i].pop(follows[i].index("e"))
    print(f'Follow({i}):',follows[i])

print('\n\n')
tterm = list(terminals)
tterm.pop(tterm.index("e"))
tterm+=["$"]
table = {}

for i in result:
    for j in tterm:
        if j in firsts[i]:
            table[(i,j)]=result[i[0]][0]
            #print(result[i[0]][0])
        else:
            table[(i,j)]=""
    if "e" in firsts[i]:
        for j in tterm:
            if j in follows[i]:
                table[(i,j)]="e"

toprint = f'{"": <10}'

for i in tterm:
    toprint+= f'|{i: <10}'
print(toprint)

for i in result:
    toprint = f'{i: <10}'
    for j in tterm:
        if table[(i,j)]!="":
            toprint+=f'|{i+"->"+table[(i,j)]: <10}'
        else:
            toprint+=f'|{table[(i,j)]: <10}'
    print(f'{"-":-<76}')
    print(toprint)
```

Output



```
Enter the number of non-terminals: 5
Enter the non-terminal: E
Enter no of terms on rhs: 1
Enter RHS term: TX
Enter the non-terminal:X
Enter no of terms on rhs: 2
Enter RHS term: +TX
Enter RHS term: e
Enter the non-terminal:T
Enter no of terms on rhs: 1
Enter RHS term: FY
Enter the non-terminal:Y
Enter no of terms on rhs: 2
Enter RHS term: *FY
Enter RHS term: e
Enter the non-terminal:F
Enter no of terms on rhs: 2
Enter RHS term: (E)
Enter RHS term: i

Dictionary of Production rules: {'E': ['TX'], 'X': ['+TX', 'e'], 'T': ['FY'], 'Y': ['*FY', 'e'], 'F': ['(E)', 'i']}


List of terminals: ['e', 'i', '(', '*', '+', ')']


First(E): ['(', 'i']
First(X): ['+', 'e']
First(T): ['(', 'i']
First(Y): ['*', 'e']
First(F): ['(', 'i']



Follow(E): ['$', ')']
Follow(X): ['$', ')']
Follow(T): ['$', '+', ')']
Follow(Y): ['$', '+', ')']
Follow(F): ['$', '*', '+', ')']
```



```
          |i        |(        |*        |+        |)        |$
----------------------------------------------------------------------
E         |E->TX    |E->TX    |         |         |         |
----------------------------------------------------------------------
X         |         |         |         |X->+TX   |X->e     |X->e
----------------------------------------------------------------------
T         |T->FY    |T->FY    |         |         |         |
----------------------------------------------------------------------
Y         |         |         |Y->*FY   |Y->e     |Y->e     |Y->e
----------------------------------------------------------------------
F         |F->(E)   |F->(E)   |         |         |         |
```