from collections import defaultdict

def compute\_first():

while True:

changed = False

for nt in rules:

for prod in rules[nt]:

tokens = prod.split()

temp = set()

for tok in tokens:

temp.update(first[tok])

if "ε" not in first[tok]:

break

else:

temp.add("ε")

if temp - first[nt]:

first[nt].update(temp)

changed = True

if not changed:

break

def compute\_follow():

follow[start\_symbol].add("$")

while True:

changed = False

for nt in rules:

for prod in rules[nt]:

tokens = prod.split()

for i, tok in enumerate(tokens):

if tok in rules:

temp = set()

for next\_tok in tokens[i+1:]:

temp.update(first[next\_tok])

if "ε" not in first[next\_tok]:

break

else:

temp.update(follow[nt])

temp.discard("ε")

if temp - follow[tok]:

follow[tok].update(temp)

changed = True

if not changed:

break

def build\_parsing\_table():

for nt in rules:

for prod in rules[nt]:

tokens = prod.split()

first\_set = set()

for tok in tokens:

first\_set.update(first[tok])

if "ε" not in first[tok]:

break

else:

first\_set.add("ε")

for terminal in first\_set - {"ε"}:

parsing\_table[nt][terminal] = prod

if "ε" in first\_set:

for terminal in follow[nt]:

parsing\_table[nt][terminal] = "ε"

def print\_parsing\_table():

print("\nLL(1) Parsing Table:")

print("-" \* (20 + len(terminals) \* 15))

print(f"{'Non-Terminal':<20}", end="")

for term in terminals:

print(f"{term:<15}", end="")

print("\n" + "-" \* (20 + len(terminals) \* 15))

for nt in non\_terminals:

print(f"{nt:<20}", end="")

for term in terminals:

print(f"{parsing\_table[nt].get(term, ''):<15}", end="")

print()

print("-" \* (20 + len(terminals) \* 15))

def parse\_string(input\_string):

stack = [start\_symbol, "$"]

input\_string += " $"

tokens = input\_string.split()

print("\nParsing Steps:")

print("-" \* 50)

print(f"{'Stack':<20}{'Input':<20}{'Action'}")

print("-" \* 50)

index = 0

while stack:

top = stack[-1]

current\_token = tokens[index]

print(f"{' '.join(stack):<20}{' '.join(tokens[index:]):<20}", end="")

if top == current\_token:

stack.pop()

index += 1

print(f"Match {current\_token}")

elif top in parsing\_table and current\_token in parsing\_table[top]:

stack.pop()

production = parsing\_table[top][current\_token]

if production != "ε":

stack.extend(reversed(production.split()))

print(f"Apply {top} -> {production}")

else:

print("Error! String rejected.")

print("\n❌ Invalid String! It does NOT belong to the grammar.")

return False

if stack == ["$"] and index == len(tokens) - 1:

print("\n✅ Valid String! It belongs to the grammar.")

return True

else:

print("\n❌ Invalid String! It does NOT belong to the grammar.")

return False

# ---------- Main Program ---------- #

rules = defaultdict(list)

first = defaultdict(set)

follow = defaultdict(set)

parsing\_table = defaultdict(dict)

# User Inputs

terminals = input("Enter terminals (space-separated): ").split()

non\_terminals = input("Enter non-terminals (space-separated): ").split()

start\_symbol = non\_terminals[0]

print("Enter productions (format: A -> alpha | beta), type 'done' to stop:")

while True:

inp = input().strip()

if inp.lower() == "done":

break

head, body = inp.split("->")

rules[head.strip()] = [p.strip() for p in body.split("|")]

# Initialize FIRST sets

for t in terminals:

first[t] = {t}

for nt in non\_terminals:

first[nt] = set()

compute\_first()

compute\_follow()

build\_parsing\_table()

print\_parsing\_table()

# Parsing Input String

input\_string = input("\nEnter input string to parse: ")

parse\_string(input\_string)