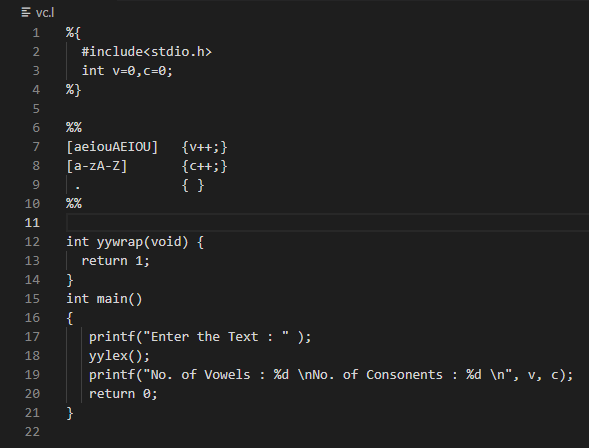
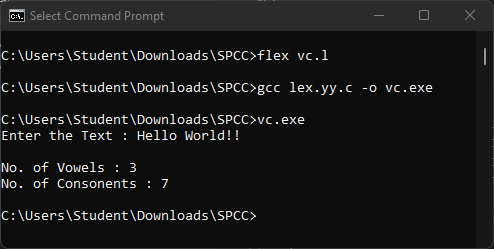
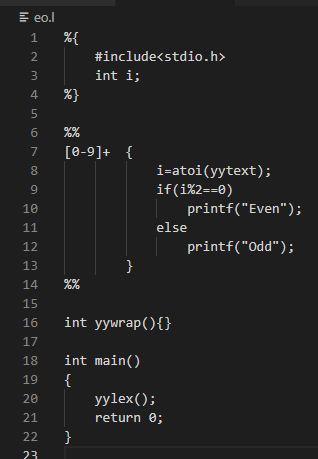
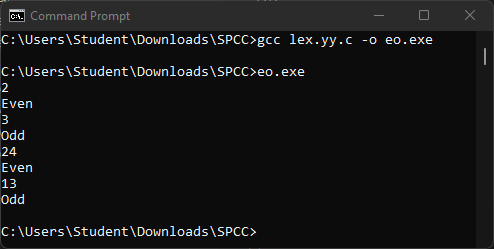


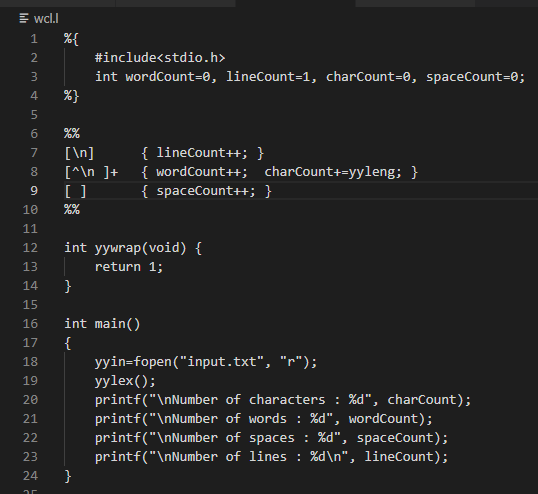
**<!-------------------------------------------------------------->**

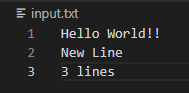


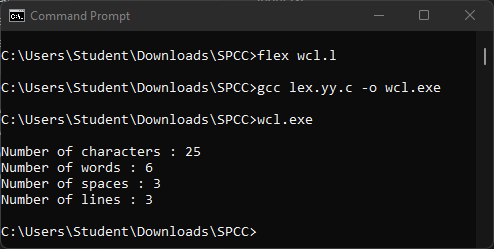


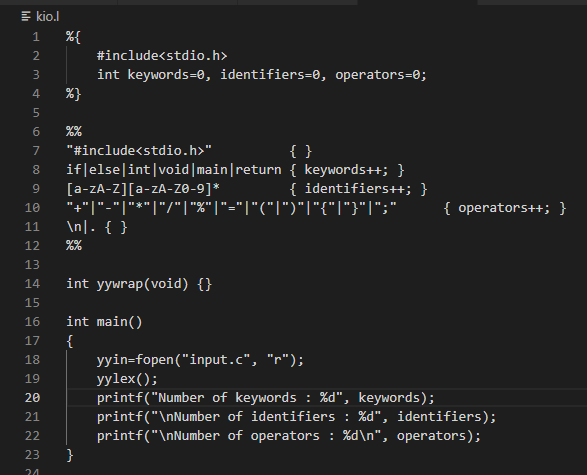


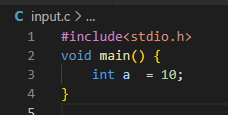


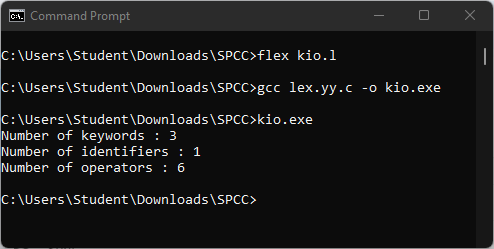


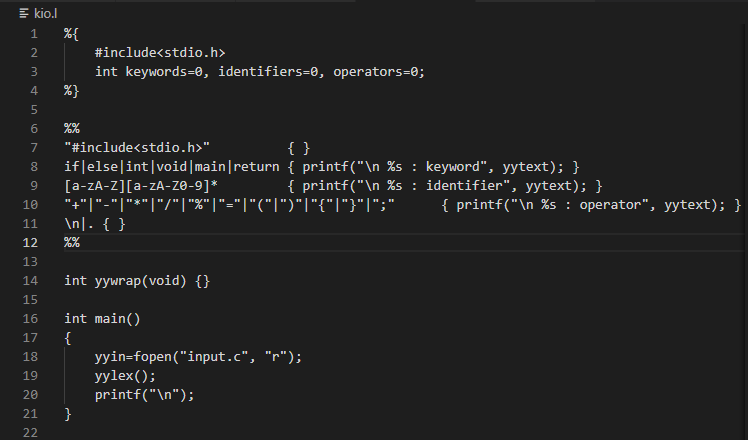


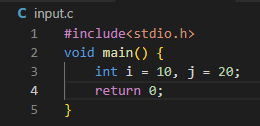


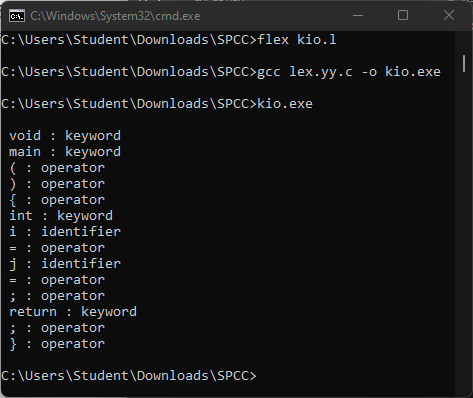


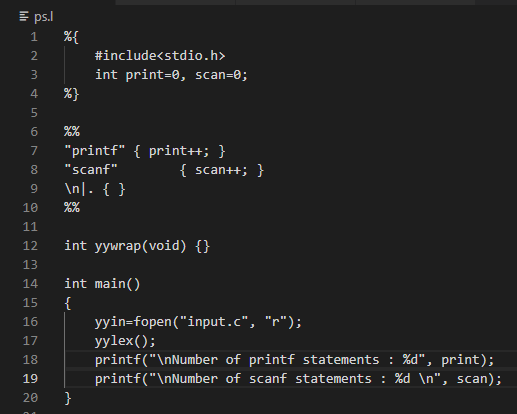


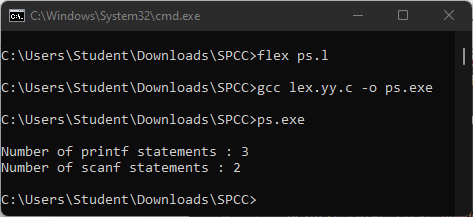
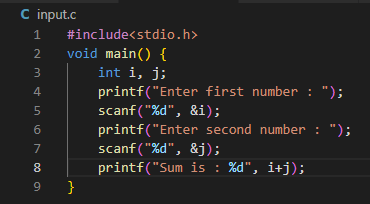


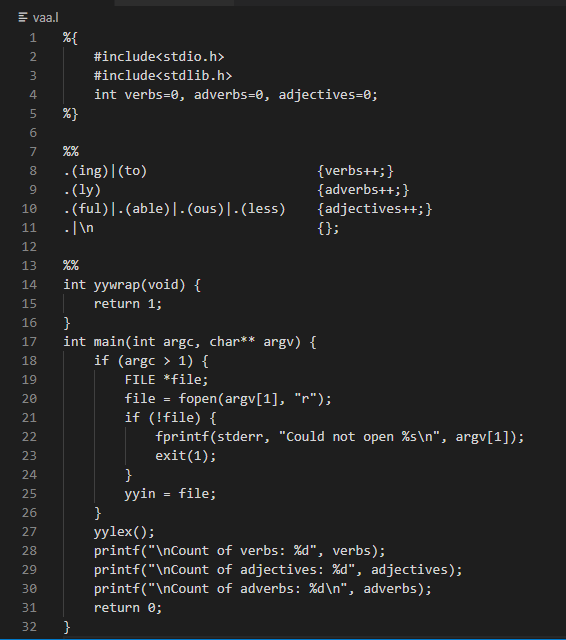


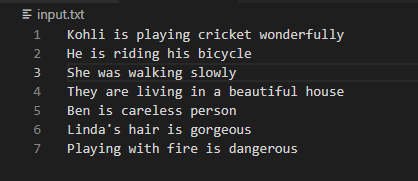


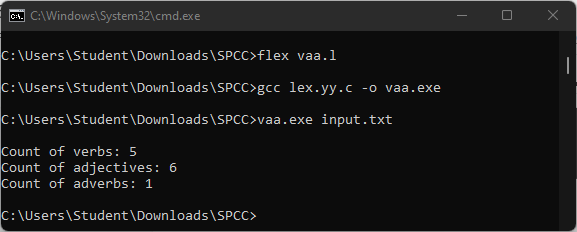












**CODE :**

from collections import OrderedDict

import re

def insert(grammar, lhs, rhs):

if(lhs in grammar and rhs not in grammar[lhs] and grammar[lhs] != "null"):

grammar[lhs].append(rhs)

elif(lhs not in grammar or grammar[lhs] == "null"):

grammar[lhs] = [rhs]

return grammar

def show\_dict(dictionary):

for key in dictionary.keys():

print(key+" : ", end = "")

for item in dictionary[key]:

if(item == epsilon):

print(f"{epsilon}, ", end = "")

else:

print(item+", ", end = "")

print("\b\b")

def first(lhs, grammar, grammar\_first):

rhs = grammar[lhs]

for i in rhs:

k = 0

flag = 0

current = []

confirm = 0

flog = 0

if(lhs in grammar and epsilon in grammar\_first[lhs]):

flog = 1

while(1):

check = []

if(k>=len(i)):

if(len(current)==0 or flag == 1 or confirm == k or flog == 1):

grammar\_first = insert(grammar\_first, lhs, epsilon)

break

if(i[k].isupper()):

if(grammar\_first[i[k]] == "null"):

grammar\_first = first(i[k], grammar, grammar\_first)

for j in grammar\_first[i[k]]:

grammar\_first = insert(grammar\_first, lhs, j)

check.append(j)

else:

grammar\_first = insert(grammar\_first, lhs, i[k])

check.append(i[k])

if(i[k]==epsilon):

flag = 1

current.extend(check)

if(epsilon not in check):

if(flog == 1):

grammar\_first = insert(grammar\_first, lhs, epsilon)

break

else:

confirm += 1

k+=1

grammar\_first[lhs].remove(epsilon)

return(grammar\_first)

def rec\_follow(k, next\_i, grammar\_follow, i, grammar, grammar\_first, lhs):

if(len(k)==next\_i):

if(grammar\_follow[i] == "null"):

grammar\_follow = follow(i, grammar, grammar\_follow)

for q in grammar\_follow[i]:

grammar\_follow = insert(grammar\_follow, lhs, q)

else:

if(k[next\_i].isupper()):

for q in grammar\_first[k[next\_i]]:

if(q==epsilon):

grammar\_follow = rec\_follow(k, next\_i+1, grammar\_follow, i, grammar, grammar\_first, lhs)

else:

grammar\_follow = insert(grammar\_follow, lhs, q)

else:

grammar\_follow = insert(grammar\_follow, lhs, k[next\_i])

return(grammar\_follow)

def follow(lhs, grammar, grammar\_follow):

for i in grammar:

j = grammar[i]

for k in j:

if(lhs in k):

next\_i = k.index(lhs)+1

grammar\_follow = rec\_follow(k, next\_i, grammar\_follow, i, grammar, grammar\_first, lhs)

if(lhs==start):

grammar\_follow = insert(grammar\_follow, lhs, "$")

return(grammar\_follow)

no\_of\_prod = int(input("Enter no. of productions : "))

separator = input("Enter Separator : ")

epsilon = input("Enter Epsilon : ")

productions = []

grammar = OrderedDict()

grammar\_first = OrderedDict()

grammar\_follow = OrderedDict()

for i in range(no\_of\_prod):

temp = input(f"Production {i+1} : ").replace(" ", "").split(separator)

item = temp[1].split('|')

for j in item:

grammar = insert(grammar, temp[0], j)

grammar\_first[temp[0]] = "null"

grammar\_follow[temp[0]] = "null"

temp[1] = temp[1].replace("|", " | ")

productions.append(f"{temp[0]} -> {temp[1]}")

start = list(grammar.keys())[0]

for lhs in grammar:

if(grammar\_first[lhs] == "null"):

grammar\_first = first(lhs, grammar, grammar\_first)

for lhs in grammar:

if(grammar\_follow[lhs] == "null"):

grammar\_follow = follow(lhs, grammar, grammar\_follow)

print("\n\nAll Proctions : ")

for i in productions:

print(i)

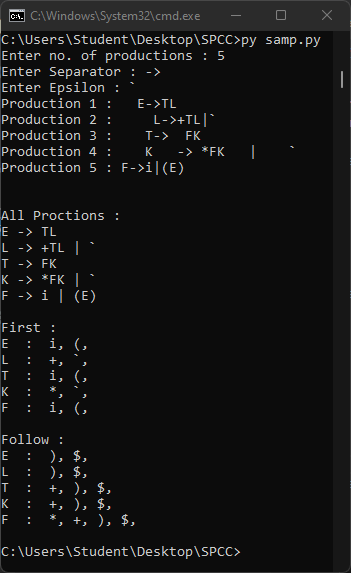
print("\nFirst : ")

show\_dict(grammar\_first)

print("\nFollow : ")

show\_dict(grammar\_follow)

**OUTPUT :**



**INPUT :**

from tabulate import tabulate

no\_of\_prod = int(input("Enter no. of productions : "))

separator = input("Enter Separator : ")

epsilon = input("Enter Epsilon : ")

startSymbol = input("Enter start symbol : ")

productions = []

terminals = []

variables = []

FIRST = {}

FOLLOW = {}

for i in range(no\_of\_prod):

temp = input(f"Production {i+1} : ").replace(" ", "").split(separator)

if temp[0].isupper() and temp[0] not in variables:

variables.append(temp[0])

productions.append(f"{temp[0]}{separator}{temp[1]}")

terminals = input("Enter the terminals : ").replace(" ", "").split(',')

for i in variables:

FIRST[i] = input(f'Enter FIRST({i}) : ').replace(" ", "").split(',')

FOLLOW[i] = input(f'Enter FOLLOW({i}) : ').replace(" ", "").split(',')

parseTable = []

temp = [''] + terminals

parseTable.append(temp)

for j in FIRST.keys():

row = ['' for x in terminals]

row = [j] + row

for i in FIRST[j]:

for k in terminals:

if (i == epsilon):

for z in FOLLOW[j]:

if z == k:

row[terminals.index(k) + 1] = f'{j} {separator} {epsilon}'

elif(i == k):

production = ''

for prod in productions:

if prod.startswith(j):

production = prod

break

temp = production.split(separator)[1].split('|')

production = temp[0]

for x in temp:

if x.startswith(i):

production = x

row[terminals.index(k) + 1] = f'{j} {separator} {production}'

parseTable.append(row)

print(tabulate(parseTable, tablefmt="simple\_grid"))

valid = True

while valid:

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

ipString = ipString + '$'

stack = '$' + startSymbol

flag = True

parseString = []

while flag:

b = stack[-1]

a = ipString[0]

if a == b == '$':

parseString[-1][2] = 'String accepted'

flag = False

elif a == b:

stack = stack[:-1]

ipString = ipString[1:]

parseString.append([stack, ipString, f'Pop {a}'])

else:

if (b == epsilon):

stack = stack[:-1]

parseString.append([stack, ipString, ''])

else:

i = variables.index(b) + 1

j = terminals.index(a) + 1

if parseTable[i][j] == '':

parseString.append([stack, ipString, 'String rejected'])

flag = False

valid = False

else:

stack = stack[:-1]

stack = stack + parseTable[i][j].replace(" ", "").split(separator)[1][::-1]

parseString.append([stack, ipString, parseTable[i][j]])

print(tabulate(parseString, tablefmt="simple\_grid"))

**OUTPUT :**

