Practical 6:

Aim: - Write a program to illustrate the generation of SPM for a given grammar.

Theory: -

Algorithm:-

1. Input the grammar from the user. Print the Terminals and Non-Terminals and Start state.
2. Obtain and print FIRST, FIRST+, LAST and LAST+ matrices and print them on the screen.
3. Compute FIRST* and LAST* and print them.
4. Calculate (±) , (є) and (϶) matrices using suitable formula. Writ the formula separately.
5. Superimpose (±) , (є) and (϶) matrices obtain SPM. (Find if It is SPG?)

Code:-

```
grammer = [["Z","bMb"],["M","(L"],['M',"a"],["L","Ma)"]]

lhs = [i[0] for i in grammer]
rhs = [i[1] for i in grammer]

#--------------------------------#
symbol = lhs + rhs
symbols = []
for i in symbol:
    for x in range(0,len(i)):
        if  i[x] not in symbols:
            symbols.append(i[x])

#symbols = ["Z","M","L","a","b","(",")"]
#--------------------------------#

def warshall(a):
    assert (len(row) == len(a) for row in a)
    n = len(a)
    for k in range(n):
        for i in range(n):
            for j in range(n):
                a[i][j] = a[i][j] or (a[i][k] and a[k][j])
    return a


def emptyMat():
    temp= []
    for i in range(0,len(symbols)):
        x = []
        for i in range(0,len(symbols)):
            x.append(0)
        temp.append(x)
    return temp

#making empty matrix
firstMatrix = emptyMat()
firstStar = emptyMat()
```

```python
I = []
#making identity matrix
identityX=0
for i in range(0,len(symbols)):
    x = []
    for j in range(0,len(symbols)):
        if j == identityX:
            x.append(1)
        else:
            x.append(0)
    identityX += 1
    I.append(x)
#making empty matrix -end


#first matrix
i = 0
for j in range(0, len(I)):
        I[i][j] = 1
        i = i+1

for i in range(0,len(lhs)):
    left = lhs[i]
    right = rhs[i]
    #first
    right = right[0]
    for i in range(0,len(symbols)):
        if symbols[i] == left:
            findL = i
            break
    for i in range(0,len(symbols)):
        if symbols[i] == right:
            findR = i
            break
    firstMatrix[findL][findR] = 1
#first matrix end

#first+ = warshal(first)
firstPlus = warshall(firstMatrix)


#--------------------------------------------------------------#

#last matrix
lastMatrix = emptyMat()
lastPlus = emptyMat()

for i in range(0,len(rhs)):
    left = lhs[i]
    right = rhs[i]
    right = right[-1]
    for i in range(0,len(symbols)):
        if symbols[i] == left:
            findL = i
            break
    for i in range(0,len(symbols)):
        if symbols[i] == right:
            findR = i
            break
    lastMatrix[findL][findR] = 1


#last+ = warshal(last)
lastPlus = warshall(lastMatrix)


#last+ transpose
lastPlusT = emptyMat()

for i in range(len(lastPlus)):
```

```
   # iterate through columns
   for j in range(len(lastPlus[0])):
       lastPlusT[j][i] = lastPlus[i][j]


#-----------------------------------------------------------------#
equal = emptyMat()

#eq matrix
#equal = resultant matrix
print("")
eqSet=[]
for i in rhs:
    if len(i) > 1:
        #ceiling function
        items = -(-len(i)//2)
        x = 0
        y = 1
        for j in range(0,items):
            temp = i[x] + i [y]
            eqSet.append(temp)
            x += 1
            y += 1

for i in eqSet:
    left = i[0]
    right = i[1]
    #print(f"left = {left}  right={right}")
    for j in range(0,len(symbols)):
        if symbols[j] == left:
            findL = j
            break

    for j in range(0,len(symbols)):
        if symbols[j] == right:
            findR = j
            break
    equal[findL][findR] = 1



#------------------------------------------------------------------#
#less then
# = eq * first+
# lessThen resultant matrix

lessThen = emptyMat()

for i in range(len(equal)):
    for j in range(len(firstPlus[0])):
        for k in range(len(firstPlus)):
            lessThen[i][j] += equal[i][k] * firstPlus[k][j]



#---------------------------------------------------------#

#first* = first+ * Identity
for i in range(0,len(firstPlus)):
    for j in range(0,len(firstPlus[0])):
        #print(f"i={i}  j={j}")
        firstStar[i][j] = firstPlus[i][j] or  I[i][j]

#--------------------------------------------------------#

#Greater then
# = last+T * eq * first*
# greaterThen resultant matrix

greaterThen = emptyMat()
```

```python
eqSfp = emptyMat()

for i in range(len(equal)):
    for j in range(len(firstStar[0])):
         for k in range(len(firstStar)):
            eqSfp[i][j] += equal[i][k] * firstStar[k][j]

for i in range(len(lastPlusT)):
    for j in range(len(eqSfp[0])):
         for k in range(len(eqSfp)):
            greaterThen[i][j] += lastPlusT[i][k] * eqSfp[k][j]

#--------------------------------------#

spm = []
for i in range(0,len(symbols)+1):
    x = []
    for i in range(0,len(symbols)+1):
        x.append(0)
    spm.append(x)
spm[0][0] = "`"

for i in range(1,len(spm)):
    spm[0][i] = symbols[i-1]
    spm[i][0] = symbols[i-1]

for i in range(1, len(lessThen)+1):
    for j in range(1, len(lessThen)+1):
        if(equal[i-1][j-1]==1):
            spm[i][j] = "="
        elif(lessThen[i-1][j-1]==1):
            spm[i][j] = "<"
        elif(greaterThen[i-1][j-1]==1):
            spm[i][j] = ">"

for i in spm:
    print ('  '.join(map(str, i)))
```

Output:-

```
`  Z  M  L  b  (  a  )
Z  0  0  0  0  0  0  0
M  0  0  0  =  0  =  0
L  0  0  0  >  0  >  0
b  0  =  0  0  <  <  0
(  0  <  =  0  <  <  0
a  0  0  0  >  0  >  =
)  0  0  0  >  0  >  0
```

Conclusion:-

We successfully constructed the simple precision matrix for the given grammar.