Nullable non-terminals
A is nullable iff a =>+ lambda

1) mark all A such that A -> lambda

2) mark all B such that B -> C1 ... Cn
   and C1 ... Cn are marked nullable

3) repeat 2 until until no more Bs can be marked

First (alpha)
let alpha = X1 ... Xn
if n = 0, return {lambda}
result <-- first[X1] - {lambda}
for ( i = 2; i<= n; i++ )
   if lambda in first[Xi-1]
      result <-- result union (first[Xi] - {lambda})
   else
      break
if (i == n+1 && lambda in first[Xn])
   result <-- result union {lambda}
return result

first[X] sets
for all a in terminals set first[a] <-- {a}
for all A in variables
   if A -> lambda is a production
      first[A] <-- {lambda}
   else
      first[A] <-- {}
for all productions of the form A -> a beta
   first[A] <-- first[A] union (a)
do
   changes <-- false
   for all productions of the form A -> B beta
      first[A] <-- first[A] union first(B beta)
   if first[A] has changed, changes <-- true
until no changes

Follow Set Algorithm
a) for A in Variables follow[A] <-- {}
b) follow[S] <-- {Lambda}
c) do
  changes <-- false
  for each production of the form A -> alpha B beta
    follow[B] <-- follow[B] union (First(beta) - {Lambda})
    if (Lambda in First(beta)) then
      follow[B] <-- follow[B] union follow[A]
      if (follow[B] has changed)
        changes <-- true
    end for
  end for
  until no changes