Nullable: It gives two results 'true' or 'false'. It is true if the empty string is a member of strings generated by the sub-expression rooted by n and false otherwise.

The construction of firstpos is made according to the following table.

	nullable(n)	firstpos(n)	lastpos(n)
leaf labelled ε	true	Φ	Φ
leaf labelled with position i	false	{i}	{i}
$+$ c_1 c_2	nullable(c_1) or nullable(c_2)	firstpos(c1) ∪ firstpos(c_2)	lastpos(c_1) ∪ lastpos(c_2)
$\bullet$ c_1 c_2	nullable(c_1) and nullable(c_2)	if (nullable(c_1)) firstpos(c_1) ∪ firstpos(c_2) else firstpos(c_1)	if (nullable(c_2)) lastpos(c_1) ∪ lastpos(c_2) else lastpos(c_2)
$*$ c_1	true	firstpos(c_1)	lastpos(c_1)

followpos is constructed only for the leaf nodes. It is constructed in the following way.

- If n is a dot (.) node containing the left child c_1 and the right child c_2, and i is a position in lastpos(c_1),then all positions in firstpos(c_2) belong to followpos(i).
- If n is a star node, and i is a position in lastpos(n), then all positions in firstpos(n) belong to followpos(i).

The DFA is constructed by the following steps:

Step I: Make the RE R as augmented by placing an end marker #, and making it $M\#$. Generate a parse tree from $M\#$.

Step II: Calculate the fi rstpos and lastpos for all the internal and leaf nodes Calculate the followpos for the leaf nodes.

Step III: Take the fi rstpos(root) as an unmarked state S of the constructing DFA.

***Step IV*:** while (there exists an unmarked state S in the states of DFA)

do

Mark S and construct a transition from S using the following process for each input symbol 'a' as an alphabet of R

do

let S contain 'a' in position $i_1, i_2..., i_n$, then

$$S^{'} = followpos(i_1) \cup ... \cup followpos(in)$$

$$\delta(S, a) = S^{/}$$

if (S' is not empty and have not appeared in the states of the DFA)

put S' as an unmarked state into the states of the DFA.