To compute FOLLOW ( A ) for all nonterminals A , apply the following rules until nothing can be added to any FOLLOW set.

1. Place $ in FOLLOW ( S ), where S is the start symbol, and $ is the input right endmarker.
2. If there is a production A → αBβ , then everything in FIRST (β) except ε is in FOLLOW ( B ).
3. If there is a production A → αB, or a production A → αBβ, where FIRST (β) contains ε, then everything in FOLLOW (A) is in FOLLOW (B).

Scala 实现 FIRST 函数：

```
def FIRST( string: ArrayBuffer[ (String, String) ] ): Map[ String, String ] = {
        val FIRST_Group = Map[ String, String ]()
        val wholeCharacters = allCharacters
        val localVT = VT
        val localVN = VN
        for( character <- wholeCharacters ) {
            // case 1
            if( localVT.contains(character) ) {
                //if there exist the original key that equals the current one
                if( FIRST_Group.contains(character.toString) == true ) {
                    val tmp = character.toString + FIRST_Group(character.toString)
                    FIRST_Group(character.toString) = tmp.distinct
                }
                //otherwise
                else {
                    FIRST_Group(character.toString) = character.toString
                }
            }
            // case 2
            if( localVN.contains(character.toString) == true ) {
                // case 2.1
                val value = findFirst(character.toString)
                if ( value.length != 0 ) {
                    if ( FIRST_Group.contains(character.toString) == true ) {
                        for( ch <- value ) {
                            val tmp = ch + FIRST_Group(character.toString)
                            FIRST_Group(character.toString) = tmp.distinct
                        }
                    }
```