We have covered one out of the two most important concepts. We will now look at the other (which is rather more complicated). The FOLLOW Set. You might have figured out what it means by its name (its name is very suggestive, just like the name “FIRST set” indicates what it might mean).  
 FOLLOW set is found ONLY FOR A NON-TERMINAL!!!!! (Actually, you could find it for a terminal as well, but by convention, it is generally defined only for non-terminal – don't ask me why, ask the authors of the Dragon Book!). This is different from FIRST, which could be calculated for any string of terminals and/or non-terminals. However, here, in case of FOLLOW, it can be computed only for **a** non-terminal, and that too NOT a string of non-terminals.   
 Very crudely, FOLLOW set for a non-terminal is the set of all the terminal symbols that can immediately come after (that is, follow) this non-terminal. For example, consider the following:  
  
S -> cAd  
A -> ab | a  
  
 Now, we make the assumption that a special marker symbol, "$" comes after the input string. Therefore, FOLLOW(S) = {$}. In other words, what follows (comes after) S is only the endmarker. But, what about FOLLOW(A)? What are all the terminal symbols that can come immediately after A? Look at the production: S -> cAd. Here, we notice that immediately after the occurrence of non-terminal A, we can only have the terminal symbol d. Thus, FOLLOW(A) = {d}. Now, you might be wondering, "Shouldn't ‘a’ also be present in FOLLOW(A)?" But no! The non-terminal A will derive the terminal a as its first symbol. It will not be followed by the terminal a. Hence, a cannot appear in FOLLOW(A).  
  
Consider:  
  
S -> cAB  
A -> ab | a  
B -> c | d  
  
Now, again, FOLLOW(S) = {$}. What is FOLLOW(A)? Ask yourself, "Whenever we get any sentential form that has non-terminal A in it somewhere, what should be the terminal symbol that comes immediately after this?" The answer to this question is the answer to FOLLOW(A)! Now, think of all possible sentential forms that have A in it somewhere. We have only one such possibility: cAB. Here, what should be the terminal symbol that comes immediately after A? Obviously, it is the very first terminal symbol that B can derive. In other words, FOLLOW(A) includes FIRST(B). Hence, we need to compute FIRST(B). What is FIRST(B)? It is the very first terminal symbol that can be derived from B. Clearly, FIRST(B) = {c, d}. Therefore, FOLLOW(A) = {c, d}. If you think about this carefully, this makes sense. If we have the sentential form: cAB, then, obviously, this sentential form can reduce to either cAc OR cAd. In other words, the only terminal symbols that can immediately follow A can only be c or d. It is clearly the first terminal symbols derivable from B.  
 What about FOLLOW(B)? Obviously, FOLLOW(B) = {$}. This is because we can see that B can only appear as the very last non-terminal symbol in any sentential form. Thus, what follows B can only be the end marker $.  
  
Next consider:  
  
S -> cAB  
A -> ab | B  
B -> c | d  
  
 Here again, FOLLOW(S) = {$} and FOLLOW(A) = FIRST(B) = {c, d}. But now, what about FOLLOW(B)? Is it only {$}? Let us consider a possible sentential form: cAB. Next, let us apply the rule: A -> B. Thus, this sentential form reduces to: cBB. If we now expand the second B, we can get: cBc or cBd. What do we notice here? We see that the immediately following terminal symbols of B are c and d. Hence, FOLLOW(B) = {c, d, $}. The $ endmarker is still there, because we can have cBB => ccB. In this case, what follows B immediately is $. Thus, what can we conclude? If a non-terminal symbol appears as the rightmost symbol in any production rule (here, B appeared as the rightmost symbol in S -> cAB as well as A -> B), then, whatever follows the LHS of the rule (in other words, the elements in the FOLLOW set of the LHS) is also there in the FOLLOW set of this non-terminal. This makes sense if you consider the above case. We had B to be the rightmost non-terminal in S -> cAB. Thus, whatever was there in FOLLOW(LHS), i.e, FOLLOW(S) was also included in FOLLOW(B). Thus, FOLLOW(B) contained $. Similarly, because B appeared as the rightmost symbol in A -> B, therefore, whatever was there in FOLLOW(LHS), i.e., FOLLOW(A) was also included in FOLLOW(B). Think of this intuitively. If at any point in the sentential form we get something like: S => cAB. Now, from the first step, we saw that $ was the immediate next symbol after S. Hence, we said that $ was in FOLLOW(S). Now that we have replaced S by something else, this something else is still followed by $. In other words, cAB is still followed by $ symbol. Thus, the rightmost non-terminal of cAB is followed by $. In other words, FOLLOW(B) includes FOLLOW(S)!  
  
Now, let us consider:  
  
S -> cABD  
A -> ab | a  
B -> c | d | lambda  
D -> e | f  
  
 Here, FOLLOW(S) = {$}. FOLLOW(D) = {$}. This is because D appears as the rightmost symbol in S -> cABD. Hence, whatever is in FOLLOW(S) (i.e., FOLLOW(LHS of the production)) is also included in FOLLOW(D). What about FOLLOW(B)? What is the first terminal symbol that immediately appears after B? We notice that in a sentential form, we can have: cABD => cABe OR cABf. Thus, we see that the terminal symbols that can immediately follow B are only e or f. Thus, FOLLOW(B) = {e, f}. What about FOLLOW(A)? Let us consider the various possible alternatives:  
  
cABD => cAcD (This is when we choose B -> c) OR  
cABD => cAdD (This is when we choose B -> d) OR  
cABD => cAD (By choosing B -> lambda) => cAe OR cAf.  
  
Thus, we notice that FOLLOW(A) = {c, d, e, f}. Why was this so? This was mainly because FIRST(B) contained lambda!!!! To find FOLLOW(A), remember what we had to do? We had to include everything in FIRST(B) in FOLLOW(A). But, because FIRST(B) contained lambda, we did not include it (it is important to note that FOLLOW set can NEVER contain lambda, unlike FIRST set). Instead we now included everything in FOLLOW(B) in FOLLOW(A)! Thus, the rule is as follows: To find FOLLOW(A), if A is not at the rightmost end of a production, then include everything in FIRST(B), where B is the symbol that immediately comes after A in the production rule. EXCEPT LAMBDA! If FIRST(B) contains lambda, then, include everything in FOLLOW(B) into FOLLOW(A). [There is actually another way to do this, rather than finding FOLLOW(A) by finding FIRST(B); you should find FIRST(beta), where beta is the entire substring that is appearing after A in the production rule (rather than just B, which is the first symbol after A in the production); include all symbols except of FIRST(beta) except lambda into FOLLOW(A); if lambda is present in FIRST(beta), then include everything in the FOLLOW(LHS of this production rule) into FOLLOW(A).]   
  
 Now, if we slightly modify the above grammar:  
  
S -> cABD  
A -> ab | a  
B -> c | d | lambda  
D -> e | f | lambda  
  
Then, FOLLOW(S) = {$}. FOLLOW(D) = {$}. FOLLOW(B) = {e, f, $}. This is because to find FOLLOW(B), we had to include everything in FIRST(D) into FOLLOW(B). But, we notice that we had lambda in FIRST(D). Therefore, we now had to include everything in FOLLOW(D) in FOLLOW(B). Thus, FOLLOW(B) contained $. Similarly, to find FOLLOW(A), we have to include everything in FIRST(B) in this. But, we notice that FIRST(B) contains lambda. Thus, we must also include FOLLOW(B) into FOLLOW(A). Therefore, FOLLOW(A) = {c, d, e, f, $}.  
  
 Thus, the rules in general for computing FOLLOW for any non-terminal are as follows (no pun intended):  
  
1. Place $ in FOLLOW(S), where S is the start symbol, and $ is the input right endmarker.  
2. If there is a production A -> (alpha)B(beta), then, everything in FIRST(beta) except lambda is in FOLLOW(B).  
3. If there is a production A -> (alpha)B OR a production A -> (alpha)B(beta), where FIRST(beta) contains lambda, then everything in FOLLOW(A) is in FOLLOW(B).  
  
In the above rules, (alpha) and (beta) can be any arbitrary string of grammar symbols, be it terminals or non-terminals.  
  
Note that for each non-terminal N that we wish to find FOLLOW of, we locate all the occurrences of N on the right hand sides of production rules!! Thus, while computing FOLLOW(N), we only look at the right hand sides of the production rules; only in special cases (rule 3) do we look at LHS of production rule! This is much different from the way we find FIRST!  
Also, note that in the third rule, if the LHS A is same as B, then, we can omit adding it, since it is meaningless (and circular) to say add everything in FOLLOW(B) into FOLLOW(B).  
  
  
**PROBLEMS:**  
  
1. Consider the following grammar. Compute FOLLOW(S) and FOLLOW(T):  
S -> aSc | T | lambda  
T -> bTc | lambda  
  
  
2. Consider the following grammar:  
E -> TE’  
E’ -> +TE’ | lambda  
T -> FT’  
T’ -> \*FT’ | lambda  
F -> (E) | i  
  
Here, the set of terminals are: { +, \*, (, ), i} and the set of non-terminals are: { E, E’, T, T’, F}. Compute the FOLLOW sets for each of the non-terminals.  
  
  
3. Consider the grammar:  
A -> BAa | lambda  
B -> bBc | AA  
Compute FOLLOW(A) and FOLLOW(B).  
  
  
4. Compute the FOLLOW sets for each of the non-terminals of the following:  
S -> aAa | bS  
A -> BB | C   
B -> bC  
C -> B | c | lambda  
  
  
5. Compute the FOLLOW sets for each of the non-terminals of the following:  
S -> LB  
B -> aSaL | bL  
E -> c | L  
J -> dEJ | f  
L -> eEJ  
The set of terminals are: {a, b, c, d, e, f} while non-terminals are: {S, B, E, J, L}  
  
  
6. Compute FOLLOW(R) and FOLLOW(T):  
T -> R | aTc  
R -> lambda | RbR  
  
  
**FOR THE ASSIGNMENT:**

Write a function named FOLLOW which takes two parameters: a grammar G and a symbol X. It should return the FOLLOW set of X for the corresponding grammar G.