Algorithm 4.6.1 — The CYK algorithm

input: context-free grammar \( G = (V, \Sigma, P, S) \)
string \( u = x_1x_2 \ldots x_n \in \Sigma^* \)

private:
\( X \): a table containing sets of variables
\( \text{step} \): the index of the “diagonal”, the main diagonal is 1, the one above it is 2, and so on.
\( i \): row index (the column index is calculated from it)
\( k \): split position in the string

// Initialize the entire table.
1. initialize all \( X_{i,j} \) to \( \emptyset \)

// Initialize the main diagonal from the rules that derive the terminals of the string.
// The main diagonal (diagonal 1) represents the length 1 substrings.
2. for \( i = 1 \) to \( n \)
   for each variable \( A \)
     if there is a rule \( A \rightarrow x_{i,i} \) then
       \( X_{i,i} := X_{i,i} \cup \{A\} \)

// Do for each “diagonal.”
// \( \text{Step} \) contains the diagonal number. Diagonal \( n \) represents the length \( n \) substrings.
3. for \( \text{step} = 2 \) to \( n \)
   // The cells start from \( i, i + \text{step} - 1 \).
   3.1 for \( i = 1 \) to \( n - \text{step} + 1 \)
     // \( i \) is the row index. It starts at 1. \( n - \text{step} + 1 \) is the last row in this diagonal.
     // For example, the diagonal 2 cells are: 1,2; 2,3; and 3,4.
     // The diagonal 3 cells are: 1,3; 2,4.
     // Do for each split position. \( k \) shows the split position.
     3.1.1 for \( k = i \) to \( i + \text{step} - 2 \)
       if there are variables \( B \in X_{i,k}, C \in X_{k+1,i+\text{step}-1} \), and a rule \( A \rightarrow BC \) then
         \( X_{i,i+\text{step}-1} = X_{i,i+\text{step}-1} \cup \{A\} \)

// If \( S \) is in the upper right corner, then the string is in the language. Otherwise, it is not.
4. if \( S \in X_{1,n} \) then
   return TRUE
else
   return FALSE