A grammar is in **Chomsky normal form** if all of its rule have one of the following forms:

1. $A \to BC$, where B, C are not the start variable.
2. $A \to a$, where a is any terminal.
3. $S \to \epsilon$.

Theorem Any context free grammar is equivalent to a grammar in Chomsky normal form.

1. Let G be a grammar in Chomsky normal form. Prove that if $w \in \Sigma^*$ can be generated by G and $|w| = n$, then it takes exactly $2n - 1$ steps to generate w using G.

2. Consider the following grammar which is in Chomsky normal form.

$$\begin{array}{llll} S \to AR & S \to a & S \to \epsilon & \\ R \to BT & R \to b & A \to a & B \to b \\ T \to CD & T \to c & C \to c & D \to d \end{array}$$

Fill in the following table by listing all variables from $V = \{S, A, B, C, D, R, T\}$ that can generate each string. For example, `c` can be generated by both C and T.

a:	ab:	abc:	abcd:
b:	bc:	bcd:	
c: C, T	cd:		
d:			

3. Let $w \in \Sigma^*$ with $|w| = n$. How many (contiguous) substrings can w have? Hint: think about the number of substrings of length 1, then length 2, etc. It might help to think about a simple example like $w =$ `abcde`.

4. The following algorithm uses dynamic programming to decided whether a string w can be generated by a grammar in Chomsky normal form. The idea is to create a table to record variables that can generate substrings of w and use it to decide if w can be generated.

```
# Build a table to track which substrings can be generated.
for substring of w:
  for k from 0 to length(w):
    left_substring = substring[0:k]
    right_substring = substring[k+1:end]
    for "A -> BC" in rules:
      if B in Table[left_substring] and C in Table[right_substring]:
        add A to Table[substring]

# Then check if w can be generated.
if Table[w] is not empty:
  return True
else:
  return False
```

What is the running time (in big-O notation) for this algorithm to decide if a string w with length n can be generated by a grammar in Chomsky normal form?