CS 635 Programming Project

One of the manipulations done in converting a CFG to Chomsky Normal Form (CNF) is to replace re-writing rules with a RHS that is not exactly two non-terminals with other rules. In what follows, small letters represent terminals, capital letters from the beginning of the alphabet (like A, B, C) and anything between <> are non-terminals. Capital letters from the end of the alphabet (like X, Y, Z) represent a symbol that may be a terminal or a non-terminal. Greek letters represent strings that may contain any mix of terminals and non-terminals.

Step 1:
For a rule A → Xβ

Split the RHS into two parts, the first symbol X, and the rest β

For each part, if it is a single non-terminal, don't change it. Other wise put <> around it, ie replace it with <X> or <β>.

Example:

A → BC unchanged
A → aB replace with A → <a>B
A → BCD replace with A → B<CD>
A → bcD replace with A → <b><cD>

Step 2:
Now, for any new non-terminals created, for a symbol of the form <Xβ> add a re-writing rule with <Xβ> as the LHS and the X and β are treated as described above to get the RHS: For both the X and β, if it is a single non-terminal, don't change it. Other wise put <> around it, ie replace it with <X> or <β>. The two resulting symbols together for the RHS. For a symbol of the form <a> add <a> → a.

Example:

for the symbol <ABC> add a rule <ABC> → A<BC>
for the symbol <aBC> add a rule <aBC> → <a><BC>
for the symbol <AB> add a rule <AB> → AB
for the symbol <a> add a rule <a> → a

Repeat step 2 for any new non-terminals created.

Complete Example:

A → aBcdE

Step 1:
A → <a><BcdE> creates 2 new non-terminals: <a> and <BcdE>

Step 2:
<a> → a
<BcdE> → B<cdE> creates new non-terminal <cdE>

repeat step 2:
<cdE> → <c><dE> creates 2 new non-terminals: <c> and <dE>

repeat step 2:
<c> → c
<dE> > <d>E creates new non-terminal <d>

repeat step 2:
<d> → d

Finished.

Write two programs, one in your choice of C++ or Java, the other in Python, to execute the above algorithm.

Input:
A → aBcdE

Output:
A → <a><BcdE>
<a> → a
<BcdE> → B<cdE>
<cdE> → <c><dE>
<c> → c
<dE> > <d>E
<d> → d