1. 

```
<expression>
  |  <term>
/  |  \  
<term> | <mult op> | <factor>
  |  |  
<factor> div ( <expr> )
  |  
<variable> | <term>
  |  |  
<identifier> | <term> | <mult op> | <factor>
  |  |  
<letter> | <factor> | * <uns const>
  |  |  
<variable> | <uns number>
  |  
<identifier> | <uns integer>
  |  
<letter> | <digit>
  |  
```

2. An equivalent grammar without useless symbols is

\[
S \rightarrow CA \\
A \rightarrow a \\
C \rightarrow b
\]

which is equivalent to the trivial grammar \( S \rightarrow ab \).

3. An equivalent essentially noncontracting grammar without chain rules is

\[
S \rightarrow bA \mid b \\
A \rightarrow Ab \mid b \mid aS \mid a \mid Ab \\
B \rightarrow aB \mid a
\]

An equivalent grammar in Chomsky Normal Form is
4. The grammar is already in Chomsky Normal Form. After the first phase of the algorithm, the grammar is

\[
S \rightarrow AB \\
A \rightarrow BS | b \\
B \rightarrow bBAZ | aZ | bBA | a \\
Z_3 \rightarrow SBA | SBAZ \\
\]

Replacing the \( A \) and the \( B \) on the righthand sides of the first two rules yields

\[
S \rightarrow bBAZ_3 | aZ | bBA | a \\
A \rightarrow bBAZ_3 | aZ | bBA | a \\
B \rightarrow bBAZ_3 | aZ | bBA | a \\
Z_3 \rightarrow SBA | SBAZ \\
\]

“Fixing” the two \( Z_3 \) productions (replacing them with ten! new rules) yields:

\[
S \rightarrow bBAZ_3 | aZ | bBA | a \\
A \rightarrow bBAZ_3 | aZ | bBA | a \\
B \rightarrow bBAZ_3 | aZ | bBA | a \\
Z_3 \rightarrow bBAZ_3 | aZ | bBA | a \\
Z_3 \rightarrow bBAZ_3 | aZ | bBA | a \\
\]

\[
S \rightarrow Y_A | b \\
Y_1 \rightarrow b \\
A \rightarrow AZ_2 | Y_3 | AY_4 | b | Y_5 | a | Y_6 | B \\
Z_2 \rightarrow Y_2 | A \\
Y_2 \rightarrow b \\
Y_3 \rightarrow b \\
Y_4 \rightarrow b \\
Y_5 \rightarrow a \\
Y_6 \rightarrow a \\
B \rightarrow Y_7 | a \\
Y_7 \rightarrow a \\
\]