1. Implement an OCaml function to print all propositional formulas of a given size, where the size measures the number of nodes in the abstract syntax tree (AST) (i.e., the number of symbols not counting parentheses). The allowable symbols are propositional variables, the unary operator \( \neg \), and the four binary operators: \( \land \), \( \lor \), \( \rightarrow \), and \( \leftrightarrow \). We are interested in all possible syntactically different formulas (even if they are tautologically equivalent), with the exception that only some combinations of variables are allowed, as described next.

For the purposes of this assignment, assume a totally ordered set of variables such as \( p_1, p_2, p_3, \) etc. Every leaf of each AST will be a propositional variable. The leaves could all be the same or they could all be different. However, in order to avoid counting equivalent formulas more than once, please only include formulas that have the following property:

\[
\text{if } p_i \text{ occurs in the formula, then for every } j < i, p_j \text{ occurs at least once to the left of } p_i.
\]

In other words, you have to use new propositional variables in order, no skipping allowed. For example, the following are formulas that should be counted:

\[
\begin{align*}
p_1 \land p_2 \\
p_1 \lor (p_2 \rightarrow p_1) \\
(p_1 \lor \neg p_1) \leftrightarrow p_1
\end{align*}
\]

The following are not:

\[
\begin{align*}
p_2 \land p_1 \\
(\neg p_1 \land p_2) \rightarrow p_4 \\
p_5 \rightarrow p_6
\end{align*}
\]

To help you check your work, the number of formulas for the first 5 sizes are: 1, 1, 9, 25, 209.

2. Using your above function, determine the number of formulas of each size for as high as you can go before it becomes computationally prohibitive (i.e., takes longer than a few minutes).

3. Plot the proportion of formulas that are tautologies against the size. Can you generate results for large enough lengths to see a trend? Is the trend as expected?