Regular grammars and finite automata

Requirement

Write a program that:
1. Reads a grammar
2. Displays the elements of a grammar, using a menu: set of non-terminals, set of terminals, set of productions, the productions of a given non-terminal symbol
3. Verifies if the grammar is regular
4. Reads the elements of a FA
5. Displays the elements of a finite automata, using a menu: the set of states, the alphabet, all the transitions, the set of final state.
6. Given a regular grammar constructs the corresponding finite automaton.
7. Given a finite automaton constructs the corresponding regular grammar.

Grammar
G = (N, E, P, S)
N = set of non-terminals
E = set of terminals
P = set of productions
S = starting symbol

Finite automaton
FA = (M, E, ρ, q0, F)
M = set of states
E = alphabet
ρ = set of transitions
qo = starting state
F = set of final states

A regular grammar is a left linear grammar (S->Aa, for any S, A from the non-terminals set and 'a' a terminal) or a right linear grammar (S->aA, for any S, A from the non-terminals set and 'a' a terminal) and if there are S->ε with S from the non-terminals set, then S does not appear in the rhs of any production.

Parsing the grammar and the finite automaton was done simply by splitting by braces, excepting the starting state / symbol.
Points 2, 5 are solved by printing previously read data.
The verification regarding the regular grammar is done by considering the following:

- if there is some S->ε (S ∈ M) then S does not appear in the right hand side of any production, otherwise it is not a RG
- if there is a production that has in the rhs a value with more than 2 symbols, it is not a RG
- if there is a production with a rhs value with 2 non-terminals, it is not a RG

- if it has both S->aA and S->Aa production types, it is not a RG
- if the rhs of a production is only a non-terminal, it is not a RG