À quelle(s) classe(s) appartiennent chacune des grammaires suivantes ? (NB : si une grammaire est de type A et $A \subset B$, cocher A et B).

a. Régulière

b. Hors Contexte

c. Ambiguë

d. Engendre un langage reconnaissable par un automate fini déterministe

e. Produit un langage non vide

1.
```
P ::= P inst ';'
    | ε
```

Correction: Linéaire à gauche, donc régulière. Elle est non ambiguë, et engendre une suite de zéro ou plusieurs `inst` *terminés* par des `;`. Ce langage est infini, régulier (puisqu'engendré par une grammaire régulière) : type 3. Il existe donc un automate, comme par exemple : `P1` engendre une liste de une ou plusieurs `inst` *séparés* par des `;`. Donc, cette grammaire engendre une liste de zéro ou plusieurs `inst` *séparés* par des `;`. Ce langage est infini, régulier (puisqu'engendré par une grammaire régulière) : type 3. Il existe donc un automate, comme par exemple :

```
                       ,----.
                       |    |
                       `----'
                        |  ^
                 `inst' |  | `;'
                        v  |
        ,----. `inst' ,----.
    --->|    |------->|    |--->
        `----'        `----'
           |
           v
```

2.
```
P  ::= P1
     | ε
P1 ::= P1 ';' P1
     | inst
```

Correction: `P1` engendre une liste de une ou plusieurs `inst` *séparés* par des `;`. Donc, cette grammaire engendre une liste de zéro ou plusieurs `inst` *séparés* par des `;`. Cette grammaire est très visiblement ambiguë. On pourrait dire que dans cette grammaire l'opérateur `;` est associatif (i.e., à droite et à gauche) : une phrase comme `inst ; inst ; inst` peut se lire comme `(inst ; inst) ; inst` ou `inst ; (inst ; inst)`. Le langage, lui, reste évidemment de type 3, et reconnu par le même automate.

```
                       ,----.
                       |    |
                       `----'
                        |  ^
                 `inst' |  | `;'
                        v  |
        ,----. `inst' ,----.
    --->|    |------->|    |--->
        `----'        `----'
           |
           v
```

3. ```
   S ::= a S b
       | ε
   ```

   Correction: $\texttt{a}^n\texttt{b}^n$... Ç'aurait été étonnant qu'il ne nous la colle pas celle-là vu comme il est créatif... Pas régulière, hors-contexte, pas ambiguë, pas reconnaissable parce que le langage $a^n b^n$ n'est pas régulier. Et oui : non $\texttt{a}^n\texttt{b}^n$ n'est pas vide.

4. ```
   S   ::= P
   P   ::= p P Q R
         | p q R
   R Q ::= Q R
   q Q ::= q q
   q R ::= q r
   r R ::= r r
   ```

   Correction: Cette grammaire est visiblement monotone, non hors contexte. Bien qu'il ne soit pas simple de le montrer formellement, une "exécution" de cette grammaire à la main montre qu'elle n'est pas ambiguë.
   On reconnaît l'exemple de grammaire engendrant $\texttt{a}^n\texttt{b}^n\texttt{c}^n$, i.e., le langage des mots commençant par un certain nombre (non nul) de `a`, puis d'autant de `b`, et enfin autant de `c`. Mais avec `p`, `q` et `r`. Ce langage est bien connu comme l'exemple type des langages sensibles au contexte (et non hors-contexte), comme vu en cours.
   Bien entendu, il est impossible de trouver un automate fini (le langage n'est pas régulier), ni même un d'automate à pile (le langage n'est pas hors-contexte).

5. Une grammaire LL(1) n'est pas ambiguë. **a.** Vrai **b.** Faux