Question I.1) *Définition inductive de la concaténation en Ocaml*

Pour toutes listes `l` et `q`, et tout élément `e` : $\begin{cases} \texttt{[]@l = l} \\ \texttt{(e::q@l = e::(q@l))} \end{cases}$

Question I.3) *associativité de la concaténation*

Soit `l2`, `l3` deux listes quelconques de meême nature.
Montrons par induction structurelle sur `l1` que pour tout `l1`, `(l1 @ l2) @ l3` = `l1 @ (l2 @ l3)`

- Si `l1 = []`, alors
- Si `l1 = e::q`, supposons que `(q @ l2) @ l3` = `q @ (l2 @ l3)` (*hypothèse d'induction*)

Donc `e::((q @ l2) @ l3)` = `e::q @ (l2 @ l3)`

Donc `((e::q) @ l2) @ l3` = `(e::q) @ (l2 @ l3)` par définition de `@`

Donc $((l1@l2)@l3) = l1@(l2@l3)$

Par induction structurelle `(l1 @ l2) @ l3` = `l1 @ (l2 @ l3)` Pour toute `l1`, donc pour toutes listes `l1`, `l2` et `3`.

Question M *o*

ntrons par induction structurelle sur `l1` que |`l1`| = |`l1`| pour toute liste `l1`.

- Si `l1` = `e::q`, On suppose que |`q`| = |`q`|.