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.2) *Rédaction de démonstration par induction structurelle*

Soit `l2` une liste. Montrons par récurrence structurelle sur `l1` que `|l1 @ l2|` = `|l1 + l2|` pour toutes listes `l1` et `l2`.

- Si `l1 = []`, `|l1 @ l2|` = `|l2|` = `|l1 + l2|`
- Si `l1` = `e::q`, supposons que `|q @ l2|` = `|q|` + `|l2|`, pour toute suite l2 (*hypothèse d'induction*) ()

$$\begin{aligned}
\texttt{|l1 @ l2|} &= \texttt{|(e::q) @ l2|} && \\
&= \texttt{|e::(q @ l2)|} && \text{par définition de } \texttt{@} \\
&= 1 + \texttt{|q @ l2|} && \text{par définition de } \texttt{l} \mapsto \texttt{l} \\
&= 1 + \texttt{|q|} + \texttt{|l2|} && \text{par hypothèse d'induction} \\
&= \texttt{|e::q|} + \texttt{|l2|} && \\
\texttt{|l1 @ l2|} &= \texttt{|l1|} + \texttt{|l2|} &&
\end{aligned}$$

Ainsi, par induction structurelle, `|l1 @ l2|` = `|l1|` + `|l2|`, pour toutes listes `l1` et `l2`, et ce, indépendamment du choix de `l2`

Question I.2)

Montrons par induction structurelle sur `l1` que pour toute liste `l1`, `|reverse(l1)|` = `|l1|`.

- Si `l1` = [], `reverse(l1)` = `reverse([])` = `[]` = `l1`
- Si `l1` = `e::q`, on suppose que `|reverse(q)|` = `|q|`

$$\begin{aligned}
\texttt{|reverse(l1)|} &= \texttt{|reverse(e::q)|} && \\
&= \texttt{|reverse(q) @ [e]|} && \text{définition de } \texttt{reverse} \\
&= \texttt{|reverse(q)|} + \texttt{|[e]|} && \\
&= \texttt{|q|} + \texttt{|[e]|} && \text{hypothèse d'induction} \\
&= \texttt{|e::q|} && \\
\texttt{|reverse(l1)|} &= \texttt{|l1|} &&
\end{aligned}$$

Ainsi, par induction structurelle, pour toute liste `l1`, `|reverse(l1)|` = `|l1|`.

Question I.3)

Montrons, par induction structurelle sur `l1`, que pour toute liste `l1`,

$$\texttt{reverse(l1 @ l2)} = \texttt{reverse(l2) @ reverse(l1)}$$

.

- Si `l1` = `[]`, alors `reverse([] @ l2)` = `reverse(l2)` = `reverse(l2) @ reverse([])`
- Si `l1` = `e::q`, supposons que `reverse(q @ l2)` = `reverse(l2) @ reverse(q)`.

$$\begin{aligned}
\text{Alors, } \texttt{reverse(l1 @ l2)} &= \texttt{reverse(e::q @ l2)} \\
&= \texttt{reverse(q @ l2) @ [e]} && \text{par définition} \\
&= \texttt{reverse(l2) @ reverse(q) @ [e]} && \text{hypothèse d'induction} \\
&= \texttt{reverse(l2) @ reverse(e::q)} && \text{par définition} \\
&= \texttt{reverse(l2) @ reverse(l1)}
\end{aligned}$$

Donc, par induction structurelle sur `l1`, $\forall$`l1`, `reverse(l1 @ l2)` = `reverse(l2) @ reverse(l1)`