TD de programmation fonctionnelle et logique

TD 4 : typage et polymorphisme

Expressions à typer

Donnée :

```
# print_int;;
- : int -> unit = <fun>
# let f x y = x;;
# let f x y = y;;
# let f x y = if true then x else y;;
# let f x y = if false then x else y;;
# let f x y = if x=y then 0 else 1;;
# let f x y = if x=y then x else 1;;
# let f x y = if x=y then x+y else x-y;;
# let f x y = if x then x+y else x-y;;
# let f x y = if x then print_int y;;
# let f g x = g(g x);;
# let rec f a b c =
   match a with
       []   -> [b c]
    | t::r  -> (b t)::(f r b c);;
# let rec f = function
     []  -> []
  | t::r -> f r;;
```

Addition polymorphe des éléments d'une liste

1. Écrivez une fonction récursive qui calcule la somme des éléments d'une liste d'entiers, en supposant que la liste est de taille supérieure ou égale à un.
2. Écrivez une fonction récursive qui calcule la somme des éléments d'une liste de taille supérieure ou égale à un, quel que soit le type des éléments de cette liste.
3. Typez la fonction précédente.
4. Écrivez une fonction récursive qui applique récursivement une opération binaire (`+`, `*.`, `^`, etc.) aux éléments d'une liste de taille supérieure ou égale à un, quel que soit le type des éléments de cette liste.
5. Même question que précédemment mais en ne supposant plus que la liste soit forcément non vide.
6. Typez la fonction précédente.