Devoir maison

Programmation fonctionnelle, ENSIIE

Semestre 4, 2019–20

1 Informations pratiques

Votre devoir maison est à déposer sur `http://exam.ensiie.fr` dans le dépôt `ipf_dm_fisa_2020` sur forme d'un PDF **avant le 4 mai à 23h59**. **Tout devoir rendu en retard se verra attribuer la note 0.**

2 Typage

Que répond OCaml sur les entrées suivantes ? Expliquer pourquoi. (Il va de soi que c'est l'explication qui m'intéresse.)

1. `let k x y = x`
2. `let s x y z = x z (y z)`
3. `let c (x, y) = y, x`
4. `let f x = x x`
5. `let rec g x = g (g x)`
6. `let h x = let i x = x in i i x`
7. `let l = [1, 2] @ [3]`

3 Listes

8. Réécrire la fonction `List.map` sans utiliser de récursion mais en utilisant `List.fold_right`.

Rappel sur l'induction sur les listes :

Pour montrer qu'une propriété $P(l)$ est vraie pour toute liste l, il suffit de montrer que :

— $P(\texttt{[]})$ est vraie ;

— pour tout élément x et toute liste q, en supposant que $P(q)$ est vraie (hypothèse d'induction), alors $P(x :: q)$ est vraie.

Dans vos démonstrations, vous indiquerez clairement quelle est la propriété $P(l)$ utilisée lors d'une preuve par induction.

9. On rappelle la définition de `append` qui concatène deux listes.

```
let rec append l1 l2 =
  match l1 with
    [] -> l2
  | x :: q -> x :: (append q l2)
```

   Démontrer que `append l []` = `l` pour toute liste `l`.

10. On définit maintenant `rev_append` de la façon suivante :

```
let rec rev_append l1 l2 =
  match l1 with
    [] -> l2
  | x :: q -> rev_append q (x :: l2)
```

   Intuitivement, `rev_append l1 l2` concatène la liste `l1` inversée à `l2`.

   Démontrer que `rev_append (rev_append l1 l2) []` = `rev_append l2 l1` pour toutes listes `l1` et `l2`.

   Indication : on pourra procéder par induction sur la liste `l1`.

11. On peut définir `rev` à l'aide de `rev_append` (c'est d'ailleurs ainsi que c'est fait dans la bibliothèque standard) :

```
let rev l = rev_append l []
```

   Démontrer que `rev (rev l)` = `l` pour toute liste `l`.