



Durée: 90 min. Le seul document autorisé est une feuille A4 blanche manuscrite

Répondre aux différentes questions en écrivant **lisiblement** à l'intérieur du cadre. Ne pas cocher les cases qui sont dans les rectangles gris, elles sont dédiées au correcteur.

Attention, les réponses écrites dans la copie double ou en dehors des cadres ne seront pas lues.

<input type="checkbox"/> 0	<input type="checkbox"/> 0	<input type="checkbox"/> 0	<input type="checkbox"/> 0	<input type="checkbox"/> 0	<input type="checkbox"/> 0
<input type="checkbox"/> 1	<input type="checkbox"/> 1	<input type="checkbox"/> 1	<input type="checkbox"/> 1	<input type="checkbox"/> 1	<input type="checkbox"/> 1
<input type="checkbox"/> 2	<input type="checkbox"/> 2	<input type="checkbox"/> 2	<input type="checkbox"/> 2	<input type="checkbox"/> 2	<input type="checkbox"/> 2
<input type="checkbox"/> 3	<input type="checkbox"/> 3	<input type="checkbox"/> 3	<input type="checkbox"/> 3	<input type="checkbox"/> 3	<input type="checkbox"/> 3
<input type="checkbox"/> 4	<input type="checkbox"/> 4	<input type="checkbox"/> 4	<input type="checkbox"/> 4	<input type="checkbox"/> 4	<input type="checkbox"/> 4
<input type="checkbox"/> 5	<input type="checkbox"/> 5	<input type="checkbox"/> 5	<input type="checkbox"/> 5	<input type="checkbox"/> 5	<input type="checkbox"/> 5
<input type="checkbox"/> 6	<input type="checkbox"/> 6	<input type="checkbox"/> 6	<input type="checkbox"/> 6	<input type="checkbox"/> 6	<input type="checkbox"/> 6
<input type="checkbox"/> 7	<input type="checkbox"/> 7	<input type="checkbox"/> 7	<input type="checkbox"/> 7	<input type="checkbox"/> 7	<input type="checkbox"/> 7
<input type="checkbox"/> 8	<input type="checkbox"/> 8	<input type="checkbox"/> 8	<input type="checkbox"/> 8	<input type="checkbox"/> 8	<input type="checkbox"/> 8
<input type="checkbox"/> 9	<input type="checkbox"/> 9	<input type="checkbox"/> 9	<input type="checkbox"/> 9	<input type="checkbox"/> 9	<input type="checkbox"/> 9

← codez le numéro de votre copie ci-contre, et écrivez à la main ce numéro ci-dessous. Sur la première page de votre copie, copier la série de 3 nombres de la forme $+x/y/z+$ figurant en haut de cette page.

Numéro de copie :

.....

1 Listes et paires

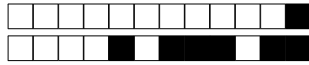
On considère la fonction `map2`. Si `l1` est une liste $[x_1; x_2; \dots; x_n]$ et `l2` une liste de même longueur $[y_1; y_2; \dots; y_n]$, alors `map2 f l1 l2` sera la liste $[f\ x_1\ y_1; f\ x_2\ y_2; \dots; f\ x_n\ y_n]$. Si `l1` et `l2` ont des longueurs différentes, alors la longueur de la liste résultat sera celle de la plus **petite** de ces deux listes.

On considère par ailleurs la fonction `zip`. Si `l1` est une liste $[x_1; x_2; \dots; x_n]$ et `l2` une liste de même longueur $[y_1; y_2; \dots; y_n]$, alors `zip l1 l2` sera la liste $[(x_1, y_1); (x_2, y_2); \dots; (x_n, y_n)]$. Si `l1` et `l2` ont des longueurs différentes, alors la longueur de la liste résultat sera celle de la plus **petite** de ces deux listes.

On considère enfin la fonction `unzip` telle que si `l1` et `l2` sont deux listes de même longueur, alors `unzip (zip l1 l1)` est égale à $(l1, l2)$.

Question 1 (0,75 pt) Donner le type de la fonction `map2`.

☐F ☐NG ☐J



Question 2 (1,25 pt) Donner le code de la fonction `map2`.

On s'interdira d'utiliser les fonctions de la bibliothèque standard OCaml (en particulier les fonctions du module `List`). Par contre, on utilisera bien les constructeurs habituels des listes OCaml.

☐E ☐M ☐R ☐mr ☐J

Question 3 (0,5 pt) Donner le type de la fonction `zip`.

☐F ☐NG ☐J

Question 4 (1,25 pt) Donner le code de la fonction `zip`.

On s'interdira d'utiliser les fonctions de la bibliothèque standard OCaml (en particulier les fonctions du module `List`) et on n'utilisera pas non plus la fonction `map2`. Par contre, on utilisera bien les constructeurs habituels des listes OCaml.

☐E ☐M ☐R ☐MR ☐J



Question 5 (0,5 pt) Donner le type de la fonction `unzip`.

☐ F ☐ NG ☐ J

Question 6 (1,25 pt) Donner le type de la fonction `unzip`.

On codera cette fonction sans utiliser directement la récursivité, mais en utilisant la fonction `List.fold_right`: `('a -> 'b -> 'b) -> 'a list -> 'b -> 'b`. On pourra utiliser les fonctions de projections des paires `fst`: `'a * 'b -> 'a` et `snd`: `'a * 'b -> 'b`.

☐ E ☐ I ☐ IFL ☐ PJ ☐ J

Question 7 (1 pt) Donner une deuxième implémentation de `map2`. Cette implémentation sera non récursive et utilisera les fonctions `List.map`: `('a -> 'b) -> 'a list -> 'b list` et `zip`.

☐ E ☐ Z ☐ M ☐ J



2 Expressions booléennes

On souhaite implémenter un évaluateur d'expression booléennes.

Une expression booléenne peut être une variable booléenne, la constante `Vrai`, la négation ("non") d'une expression booléenne ou encore la combinaison de deux expressions booléennes via une conjonction ("et"). Remarque: Pour simplifier, on ne gèrera pas de constante "faux", ni le "ou" dans cet exercice. Cela n'est pas restrictif car on peut les exprimer via les autres constructions. On appellera `boolexpr` le type OCaml représentant ces expressions booléennes.

On veut pouvoir écrire les expressions booléennes suivantes, données à titre d'exemple:

- `Var("x")`
- `Non(Vrai)`
- `Et(Vrai, Var("x"))`
- `Et(Non(Var("x")), Et(Var("y"), Vrai))`

Le résultat de l'évaluation d'une expression booléenne sera de type `bool option`. La valeur d'une variable booléenne sera de type `bool`.

Les valeurs des variables seront placées dans un *environnement*. Un environnement est une liste d'association qui fait correspondre au nom d'une variable la valeur de celle-ci. Un environnement aura donc le type `(string * bool) list`.

On rappelle que la fonction `List.assoc_opt: 'a -> ('a * 'b) list -> 'b option` prend une clé et une liste d'association et renvoie `Some v` si `v` est la valeur associée à la clé dans la liste et renvoie `None` si aucune valeur n'est associée à la clé dans la liste.

On considère la fonction `eval` qui prend un environnement et une expression booléenne (dans cet ordre) et renvoie le résultat de l'évaluation de cette expression dans l'environnement fourni.

Question 8 (2 pt) Définir le type `boolexpr`.

☐ F ☐ M2T ☐ M1T ☐ J



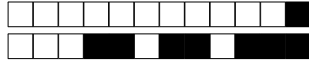
+1/5/56+

Question 9 (1 pt) Donner le type de la fonction `eval`.

☐ F ☐ J

Question 10 (4 pt) Donner le code de la fonction `eval`.

☐ F ☐ RC ☐ LC ☐ RLC ☐ EM ☐ J



3 Modules et arbres de recherche

On souhaite coder une structure de donnée d'arbre binaire pour coder des ensembles. Afin de rendre cette structure adaptable au type des éléments de l'ensemble, l'implémentation se basera sur des foncteurs.

On considère la signature suivante:

```
module type Cmp = sig
  type t
  type cmp_t = Lt | Gt | Eq
  val cmp : t -> t -> cmp_t
end
```

Les modules d'arbres binaires de recherche devront comporter un type **ens** pour représenter les ensembles (qui sont codés par les arbres binaires de recherche), un type **elt** pour représenter les éléments des ensembles, et deux fonctions. La première est **appartient**, qui prend un élément et un ensemble et renvoie un booléen indiquant si l'élément est dans l'ensemble. La deuxième est **insere** qui ajoute un élément dans un ensemble.

On considère le code suivant à compléter:

```
module ABR_T (M: Cmp): (ENS_T with type elt = M.t) = struct
  type ens = (* à compléter *)
  type elt = M.t
  let rec appartient v a = (* à compléter *)
  let rec insere v a = (* à compléter *)
end
```

Question 11 (1,5 pt) Donner la signature ENS_T

☐ F ☐ TC ☐ VC ☐ J



+1/7/54+

Question 12 (1 pt) Compléter le type `ens`

☐ F ☐ CC ☐ DC ☐ J

Question 13 (2 pt) Compléter la fonction `appartient`

☐ F ☐ AC ☐ MC ☐ J



+1/8/53+

Question 14 (2 pt) Compléter la fonction insere

☐ F ☐ AC ☐ MC ☐ J