TME6 – Ensembles d'entiers

Exercice 6.1 (Extrait Examen Janvier 2023)**.**
Dans cet exercice, on pourra utiliser les fonctions définies dans l'exercice **??** du TME2.

Un arbre binaire est dit *équilibré* si tous les chemin (à partir de la racine jusqu'à une feuille) sont de même longueur.

Exemple : l'arbre ci-dessous à gauche est équilibré, mais l'arbre ci-dessous à droite ne l'est pas (on représente les nœuds internes par des ○, et les feuilles par des □).

Notez qu'un arbre équilibré de hauteur k a exactement 2^k feuilles.

Dans cet exercice, la hauteur $h(t)$ d'un arbre binaire t est 0 si t est une feuille, et $1+\max(h(g), h(d))$ si t est un arbre avec un fils gauche g et un fils droit d. Si t est un arbre binaire équilibré alors $h(g) = h(d)$, et dans ce cas $h(t) = 1 + h(g) = 1 + h(d)$.

On souhaite représenter un ensemble d'entiers par un arbre binaire équilibré dont les nœuds internes ne sont pas étiquetés et les feuilles sont étiquetées par des booléens.

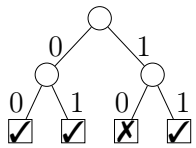
Un arbre binaire équilibré de hauteur k représente le sous-ensemble des entiers de $\{0, \ldots, 2^k - 1\}$ dont une représentation binaire inversée représente un chemin de la racine à une feuille marquée `true` tel qu'à chaque étape du cheminement, le bit 0 indique de descendre sur le sous-arbre gauche tandis que le bit 1 indique de descendre sur le sous-arbre droit.

Exemple : Dans l'arbre ci-contre ✓ représente le booléen `true`, ✗ représente le booléen `false`. Cet arbre représente l'ensemble $\{0, 2, 3\}$ puisque 00, 01, et 11 sont les représentations binaires inversées des entiers 0, 2 et 3. Il s'agit bien d'un sous ensemble de $\{0, 1, 2, 3\}$.

Afin de représenter de tels arbres en OCaml on définit le type `int_set` ci-contre. L'arbre de l'exemple ci-dessus est représenté par la valeur OCaml `ex` définie ci-contre. Les bits 0 et 1 qui figurent sur les arêtes correspondent aux indications de chemin (0 pour l'arête gauche et 1 pour l'arête droite) mais ne font pas partie de la représentation des arbres.

On suppose définies les exceptions `Not_perfect` et `No_fit`.

```
(* Un arbre représentant
   l'ensemble {0;2;3} *)
```




```
type int_set =
  | N of int_set * int_set
  | E of bool

let ex =
    N(N(E true,
        E true),
      N(E false,
        E true))

exception Not_perfect
exception No_fit
```

1. Définir une fonction de signature `hauteur_equilibre (t: int_set): int` qui calcule la hauteur de l'arbre `t` si `t` est un arbre équilibré et lève l'exception `Not_perfect` sinon. Attention l'arbre `E(...)` est de hauteur 0.

   Exemples :
   `hauteur_equilibre ex = 2`
   `hauteur_equilibre (N(N(E true,E true),E false))` lève l'exception `Not_perfect`

2. Définir une fonction de signature `mem (x: int) (t: int_set): bool` permettant de tester si l'entier `x` appartient à l'ensemble représenté par l'arbre `t`.

   Indication : l'entier `x` appartient à l'ensemble représenté par l'arbre `t` si la liste correspondant à une représentation binaire inversée de `x` est un chemin de la racine de `t`

   — à une feuille étiquetée par le booléen `true`

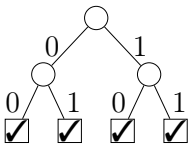
   — ou bien à un nœud interne n à partir duquel on accède à une feuille étiquetée par `0` en se déplaçant uniquement vers la gauche depuis le nœud n (puisque si l est une liste de booléens représentant un entier, alors la liste obtenue en ajoutant un nombre quelconque `0` à la fin de la liste l représente le même entier)

   La fonction `mem` pourra donc utiliser une fonction auxiliaire dont le paramètre est la représentation binaire inversée du paramètre `n` de `mem`.

   Exemples :
   `mem 2 ex` = `true`
   `mem 1 ex` = `false`

3. Définir une fonction de signature `insert_exn (t: int_set) (x: int):int_set` qui construit l'arbre représentant l'ensemble des entiers $F_t \cup \{x\}$ où F_t est l'ensemble des entiers représenté par `t` ; l'arbre construit doit avoir la même hauteur que l'arbre `t`. Si `x` ne peut être inséré dans `t` sans changer la hauteur de `t`, alors la fonction `insert_exn` lèvera l'exception `No_fit`. L'insertion n'est donc possible que si elle peut se faire uniquement en changeant l'étiquette d'une feuille.

   Exemples :
   `insert_exn ex 1` =

   `(insert_exn ex 25)` lève l'exception `No_fit`

   Indication : on pourra calculer la liste correspondant à une représentation binaire inversée de `x`, et suivre le chemin de la racine de `t` jusqu'à une feuille comme dans la question précédente (c'est-à-dire en se déplaçant à gauche dans l'arbre lorsque la liste est vide).

4. Définir une fonction de signature `increase_height (t: int_set) : int_set` qui construit un arbre de hauteur $h(t) + 1$ et qui représente exactement le même ensemble d'entiers que l'arbre `t`.

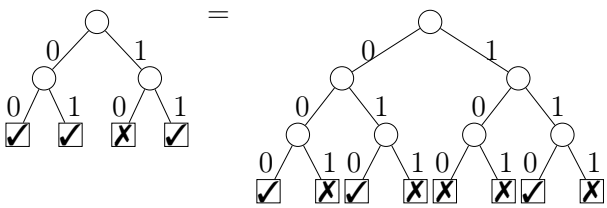
   Indication : il s'agit d'ajouter un fils gauche et un fils droit à chaque feuille de l'arbre `t`

   — si cette feuille est étiquetée par le booléen `true` alors l'entier correspondant au chemin menant à cette feuille appartient à l'ensemble représenté par `t` et il suffit alors que le fils gauche ajouté soit étiqueté par le booléen `true` (puisqu'un déplacement vers la gauche correspond au bit `0` sur le chemin et qu'ajouter `0` à la fin d'un chemin ne change pas la valeur de l'entier représenté par le chemin) et que le fils droit ajouté soit étiqueté par le booléen `false` (pour ne pas ajouter d'entier à l'ensemble représenté par `t`)

   — si cette feuille est étiquetée par le booléen `false` alors l'entier correspondant au chemin menant à cette feuille n'appartient pas à l'ensemble représenté par `t` et il suffit alors que les fils gauche et droit ajoutés soient étiquetés par le booléen `false` (pour ne pas ajouter d'entier à l'ensemble représenté par `t`)

   Exemple :

`increase_height` =



5. Déduire des deux questions précédentes une fonction de signature :

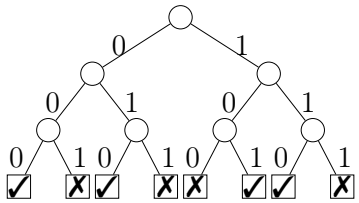
   ```
   insert (x: int) (t: int_set) : int_set
   ```

   qui construit l'arbre représentant l'ensemble des entiers $F_t \cup \{\texttt{x}\}$ où F_t est l'ensemble des entiers représenté par `t`. La taille de l'arbre sera augmentée au besoin, autant de fois que nécessaire.

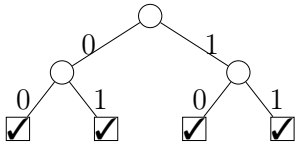
   Indication : il suffit d'essayer d'ajouter l'entier `x` dans l'ensemble représenté par `t`, et si cette tentative échoue, d'essayer à nouveau sur l'arbre `t` dont la hauteur a été augmentée de 1, et ainsi de suite.

   Exemples :

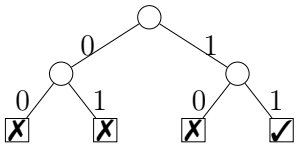
   `insert 5 ex` =



   `insert 1 ex` =



   `insert 3 (E false)` =



6. *En déduire* une fonction de signature `list_to_tree (l: int list) : int_set` qui construit l'arbre représentant l'ensemble des entiers contenus dans la liste `l`.

   Exemples :

   ```
   list_to_tree [] = E false
   list_to_tree [0;2;3] = ex
   ```

7. Définir une fonction de signature :

   ```
   path_list (t : int_set) : ((int list) * bool) list
   ```

   permettant de construire la liste de toutes les paires `(c,b)` telles que `c` est une liste correspondant à un chemin de la racine à une feuille de l'arbre `t` et `b` est le booléen qui étiquette cette feuille.

   Exemple :

   ```
   path_list ex = [([0; 0], true); ([0; 1], true); ([1; 0], false); ([1; 1], true)]
   ```

8. En déduire une fonction de signature :

   ```
   tree_to_list (t : int_set) : int list
   ```

   permettant de construire la liste des entiers appartenant à l'ensemble représenté par l'arbre `t`.

   Exemple :

   ```
   tree_to_list ex = [0; 2; 3]
   ```