

I Base de données d'épidémies

1. Aucun attribut ne peut être clé primaire (le nom « Brésil » apparaît plusieurs fois, de même que l'iso « BR » et l'année « 2010 »).
Par contre le couple (nom, annee) est une clé primaire.

2.

```
SELECT * FROM palu
WHERE annee = 2010 AND deces >= 1000
```

3.

```
SELECT 100000*cas/pop
FROM palu
JOIN demographie
ON iso = pays AND annee = periode
WHERE annee = 2011
```

II Base de donnée de corps célestes

A `SELECT masse FROM CORPS`

B.1

```
SELECT COUNT(DISTINCT id_corps)
FROM etat
WHERE datem < tmin()
```

B.2

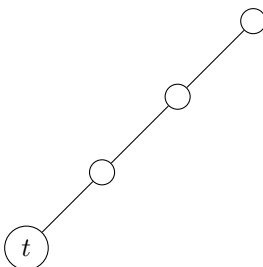
```
SELECT id_corps, MAX(datem)
FROM etat
WHERE datem < tmin()
GROUP BY id_corps
```

B.3

```
SELECT masse,x,y,z,vx,vy,vz FROM corps AS c
JOIN etat AS e ON c.id_corps = e.id_corps
JOIN date_mesure AS d ON (datem = date_der AND e.id_corps = d.id_corps)
WHERE masse >= masse_min() AND ABS(x) < arete()/2
AND ABS(y) < arete()/2 AND ABS(z) < arete()/2
ORDER BY x*x+y*y+z*z
```

III Programmation fonctionnelle

1. `height` est appelé exactement une fois sur chaque noeud `N` et sur chaque `E`. Donc le nombre total d'appels à `height` dans le calcul de `height (N (N (E, N (E, E)), N (E, E)))` est 9.
2. `left` est de type `tree -> int -> tree` et `left t n` construit un arbre de la forme suivante (peigne gauche), avec `n` noeuds au dessus de `t` :



3. Lors qu'une fonction **f** effectue un autre appel de fonction **g** (qui peut être un appel récursif), l'exécution de **f** se met en pause et la position actuelle dans le code est empilé sur la *call stack* (pile d'appels). Ceci est nécessaire pour que le processeur sache à quel endroit reprendre l'exécution de **f**, une fois l'exécution de **g** terminée. Comme empiler dans la *call stack* demande de l'espace mémoire, le nombre d'appels que l'on peut y stocker est limité, d'où le message *StackOverflow* lorsque qu'elle saturée.

C'est ce qui arrive avec l'appel **height t**.

Par contre, **left** est une fonction récursive terminale qui est automatiquement transformée en boucle par le compilateur et qui n'utilise donc pas la *call stack*.

4.

```

let rec aux1 t k =
  match t with
  | E -> k 0
  | N (l, r) ->
    aux1 l ((*1*) fun hl ->
      aux1 r ((*2*) fun hr ->
        k (1 + max hl hr)))

let height1 t =
  aux1 t ((*3*) fun h -> h)

```

Dans l'exécution détaillée ci-dessous, on a utilisé **id** pour **fun h -> h**. Chaque flèche correspond à un appel de fonction. À droite est noté le prochain appel de fonction effectué.

height1 (N(E, N(E, E)))	height1
→ aux1 (N(E, N(E, E))) id	aux1
→ aux1 E (fun hl -> aux1 (N(E, E)) (fun hr -> id (1 + max hl hr)))	aux1
→ (fun hl -> aux1 (N(E, E)) (fun hr -> id (1 + max hl hr))) 0	(*1*)
→ aux1 (N(E, E)) (fun hr -> id (1 + max 0 hr)) = aux1 (N(E, E)) (fun hr -> id (1 + max 0 hr))	aux1
→ aux1 E (fun hl -> aux1 E (fun hr -> (fun hr -> id (1 + max 0 hr)) (1 + max hl hr)))	aux1
→ (fun hl -> aux1 E (fun hr -> (fun hr -> id (1 + max 0 hr)) (1 + max hl hr))) 0	(*1*)
→ aux1 E (fun hr -> (fun hr -> id (1 + max 0 hr)) (1 + max 0 hr))	aux1
→ (fun hr -> (fun hr -> id (1 + max 0 hr)) (1 + max 0 hr)) 0	(*2*)
→ (fun hr -> id (1 + max 0 hr)) (1 + max 0 0)	(*2*)
→ id (1 max 0 (1 + max 0 0))	(*3*)
→ 2	

5. **aux1** est de type **tree -> (int -> 'a) -> 'a**

6. Montrons, par induction structurelle :

$H(t)$: Pour toute fonction $k : \text{int} \rightarrow \text{int}$, **aux1 t k** renvoie $k(h(t))$ (où $h(t)$ est la hauteur de **t**).

- $H(E)$ est vraie car **aux1 E k** → **k 0** et $h(t) = 0$.

- Supposons $H(l)$ et $H(r)$.

aux1 (N(l, r)) k → **aux1 l (fun hl -> aux1 r (fun hr -> k (1 + max hl hr)))**

D'après $H(r)$, **aux1 r (fun hr -> k (1 + max hl hr))** renvoie **k (1 + max hl hr')**, où **hr'** est la hauteur de **r**.
D'après $H(l)$, **aux1 l (fun hl -> aux1 r (fun hr -> k (1 + max hl hr)))** renvoie donc **k (1 + max hl' hr')**, où **hl'** est la hauteur de **l**.

Par définition de la hauteur, **aux1 (N(l, r)) k** renvoie donc bien **k h**, où **h** est la hauteur de **N(l, r)**, ce qui termine de prouver l'induction.

7.

8.

```
type cont =
  | K1 of tree * cont
  | K2 of int * cont
  | K3

let rec aux2 t k =
  match t with
  | E -> apply k 0
  | N (l, r) -> aux2 l (K1(r, k))
and apply k v =
  match k with
  | K1 (r, k) -> aux2 r (K2(v, k))
  | K2 (h, k) -> apply k (1 + if v > h then v else h)
  | K3 -> v

let height2 t =
  aux2 t K3
```
