Dictionnaire

Un dictionnaire est un type abstrait particulièrement simple, car il ne permet que trois opérations : insérer un objet, supprimer un objet, et déterminer si un objet est membre du dictionnaire. Un dictionnaire peut par exemple être utilisé pour les membres d'une association, pour les clients d'une société, pour les fichiers d'un répertoire, *etc.* Pour que ce soit plus utile, on associe une clé unique avec chaque objet, par exemple, le numéro de sécurité sociale, le numéro de téléphone, ou le nom du fichier. Les clés sont souvent des objets relativement simple, comme un nombre ou une chaîne de caractères.

Dans ce TD, nous allons implanter le type abstrait dictionnaire en utilisant le type concret arbre binaire de recherche. D'une manière générale, un arbre de recherche est un arbre utilisé pour stocker dans ses noeuds des données d'un domaine ordonné de manière à ce que la donnée la plus petite (selon l'ordre du domaine) soit stockée dans le noeud le plus à gauche, et la donnée la plus grande celui le plus à droite.

Dans le cas particulier d'un arbre binaire, le sous-arbre de gauche d'un noeud contient des données strictement inférieures à celles stockées dans le noeud, alors que le sous-arbre de droite contient des données strictement supérieures à celle du noeud.

Interface. Voilà l'interface C que nous proposons pour une file (fichier `map.h`).

```
#ifndef MAP_H
#define MAP_H
/* Header file for the map abstract data type (map.h) */

struct map_t;

typedef struct map_t *map;

/* map an integer key to an object pointer */
typedef int (*keyfunc)(void *);

/* create an empty map */
map map_create(keyfunc f);

/* create an empty map */
void map_destroy(map f);

/* create an empty map */
int map_height(map f);

/* find an object in the map and return it or NULL if not found */
void * map_find(map m, int key);

/* insert an object in a map and return it or NULL in case of
   failure (NULL object or object already inside the map) */
void * map_insert(map m, void * object);
```

```
/* delete an object from a map and return it or NULL in not found */
void * map_delete(map m, int key);

/* dump the underlying binary research tree */
void map_dump(map m);

#endif
```

1 Implémentation récursive d'un dictionnaire par un arbre binaire de recherche

1. Donner une première implémentation récursive un dictionnaire par un arbre binaire de recherche.

2 Implémentation itérative d'un dictionnaire par un arbre binaire de recherche

1. Donner une implémentation itérative un dictionnaire par un arbre binaire de recherche. Vous pouvez modifier la structure `tree_t`.

3 Implémentation récursive terminale d'un dictionnaire avec une liste chaînée

1. Modifier votre première implémentation récursive pour qu'elle devienne récursive terminale.