TD n°10 - Structure de données sac

Dans ce TD on considère une structure de données appelée "sac". On y stocke des données de manière non ordonnée, avec doublons, pas forcément séquentielle.

Les primitives pour un sac d'entiers sont les suivantes, on les exprime en Ocaml car on étudiera une implémentation en Ocaml :

- `creer : void -> sac` qui crée un sac vide.
- `est_vide : sac -> bool` qui teste si le sac est vide.
- `prendre : sac -> int` qui prend un objet dans le sac (sans précision de lequel).
- `mettre : sac -> int -> unit` qui met un objet dans le sac.

On considère que la structure de données est mutable et qu'on a en OCaml le type `sac` qui représente un sac d'entiers et toutes les primitives préimplémentées.

1. Écrire une fonction `exemple_sac : unit -> sac` qui construit en C un sac contenant les valeurs 0,0,1,2,4,5,10 et 23.
2. Écrire une fonction `taille : sac -> int` qui calcule la taille d'un sac. **Attention :** à la fin de la fonction, le sac doit être le même qu'au début.

On va maintenant implémenter le sac en Ocaml.

3. Comment faire une implémentation mutable de sac en Ocaml ?
4. Implémenter les primitives. C'est à vous de réfléchir aux détails de l'implémentation.
5. Écrire une fonction `taille2 : sac -> int` qui calcule la taille d'un sac, cette fois-ci en sachant comment le sac a été implémenté.
6. Comparer la complexité des deux fonctions `taille` et `taille2`.

Si vous avez terminé, passer en C et écrire des fonctions sur les listes chainées qui permettent de renvoyer l'élément i, insérer un élément à l'indice i, supprimer l'élément i, tester l'égalité entre deux listes et renverser la liste, d'abord en utilisant que les primitives, puis en utilisant votre connaissance de l'implémentation.