TD n°6 : Gestion mémoire et structures autoréférentes

Exercice 1 (assimilation)

Vous devez implanter une liste d'entiers doublement chaînée. cette liste sera représentée par deux structures : **listeInt**, qui contient un pointeur sur le premier maillon et un pointeur sur un maillon appelé maillon courant, et **maillon**, qui contient un entier (la valeur stockée par le maillon), un pointeur sur le maillon précédent (si applicable) et un pointeur sur le maillon suivant (si applicable).

Vous devez réaliser les fonctions suivantes :

- **void initListe(listeInt* w)** : crée une liste vide.
- **void ajouteFin(listeInt* w, int x)** : ajoute un entier x à la fin de la liste.
- **void razCurseur(listeInt* w)** : remet le curseur en début de liste.
- **void deplaceCurseur(listeInt* w, int d)** : déplace le curseur de d position vers l'avant (si d est positif) ou l'arrière (si x est négatif) sans dépasser les limites de la liste.
- **int valCurseur(listeInt* w)** : retourne l'entier situé dans la position indiquée par le curseur.

Exercice 2 (consolidation)

Reprenez l'exercice 1 en ajoutant les fonctions suivantes :

- **void insereVal(listeInt* w, int x)** : insère une valeur à la position du curseur.
- **void supprimeVal(listeInt* w)** : supprime la valeur située à la position du curseur.
- **void echangeVals(listeInt* w)** : échange les valeurs situées à la position du curseur et à la position suivante.

Exercice 3 (assimilation)

Un arbre binaire est soit vide, soit constitué d'une nœud racine auquel sont rattachés un fils droit et un fils gauche, qui sont des arbres binaires. On s'intéresse ici aux arbres étiqueté par des entiers, c'est à dire dont chaque nœud stocke à une valeur entière. Vous devez proposer une structure **arbreBin** permettant de représenter de tels arbres binaires et réaliser les fonctions suivantes:

- **void affichePost(arbreBin* a)** : affiche les valeurs stockées dans les nœuds en ordre postfixe.
- **int profondeur(arbreBin* a)** : retourne la profondeur de l'arbre, c'est à dire le nombre de nœuds du plus long chemin partant de la racine.

Exercice 4 (consolidation)

On dit qu'un arbre binaire est trié si son fils gauche et son fils droit sont des arbres triés et si toutes les valeurs stockées dans le fils gauches sont inférieures ou égales à la valeur stockée dans le nœud racine et toutes les valeurs stockées dans le fils droit sont strictement supérieures. Reprenez l'exercice 3 ajoutant les fonctions suivantes :

- **arbreBin* ajouteVal(arbreBin* a, int val)** : Retourne l'arbre binaire trié obtenu en insérant la valeur **val** dans d'arbre trié pointé par **a**.
- **int estDans(arbreBin* a, int val)** : Vérifie si l'entier **val** est dans l'arbre trié **a**.
- **arbreBin* retireVal(arbreBin* a, int val)** : Retourne un arbre binaire trié obtenu en retirant la valeur **val** de d'arbre trié pointé par **a**. On supposera que cette valeur a une seule occurrence dans l'arbre initial.