TD n°7 : Structures chaînées et autoréférentes en C++

Exercice 1 (assimilation)

Vous devez réaliser une classe **ListInt** représentant une liste d'entiers avec un modèle "tête queue".

Les variables d'instance privées de cette classe sont les suivantes.

```
class ListInt
{
    private:
    bool vide;
    int tete;
    ListInt* queue;
    //...
};
```

Dotez cette classe d'un constructeur par défaut créant une liste vide et d'un constructeur **ListInt(int tete, ListInt* queue)** qui crée une liste en connectant un élément de tête à une liste existante dont on donne l'adresse. Dotez là également d'une méthode publique **affiche()** qui affiche les valeurs contenues dans une liste.

Dessinez les représentations en mémoire des listes construites avec le code présenté ci-contre.

```
ListInt w1(11,new ListInt(22,new ListInt()));
ListInt w2(55,&w1);
```

Dotez la classe **ListInt** d'un destructeur qui détruit complètement une liste et restitue au système toute la mémoire qu'elle occupait. A la fin de l'exécution de la méthode dans laquelle le code ci-dessus a été définit, ce destructeur sera appelé automatiquement (car w1 et w2 sont des méthodes locales de cette méthode et doivent donc être détruites à l'issue de son exécution). A ce moment là, on constate parfois un plantage du programme. Expliquez pourquoi et précisez les précautions à prendre pour éviter ce problème.

Exercice 2 (consolidation)

Vous devez maintenant doter la classe **ListInt** d'un nouveau constructeur **ListInt(int tete, ListInt queue)** qui crée une liste ayant pour tête la valeur **tete** et pour queue une *copie* de la liste représentée par le paramètre **queue**. A cette fin, vous devez réaliser une méthode privée clone() qui produit un **ListInt* clone()** d'une liste, c'est à dire une copie ne partageant aucune mémoire avec l'original.

Dessinez les représentations en mémoire des listes construites avec le code présenté ci-contre.

```
ListInt w1(11,ListInt(22,ListInt()));
ListInt w2(55,w1);
```

Pour chacune des lignes ci dessous, combien d'instances de la classe ListInt sont créées ? N'oubliez pas les appels réalisés par la fonction clone. Vous devez aussi compter, si applicable, les instances créées temporairement et détruites ensuite.

```
ListInt w1(11,ListInt(22,ListInt(33,ListInt())));
ListInt w2(11,new ListInt(22,new ListInt(33,new ListInt())));
```

Ajoutez à la classe **ListInt** un constructeur en copie **ListInt(ListInt& w)** et une méthode de surcharge de l'opérateur d'affectation **ListInt& operator=(const ListInt& a)**. Ces méthodes doivent faire en sorte qu'en cas d'affectation ou d'initialisation d'une variable avec un liste existante, une copie de la liste existante soit créée.

TD n°7 : Structures chaînées et autoréférentes en C++

Exercice 3 (consolidation)

Dessinez les représentations en mémoire des listes créées par le code ci-dessous.

```
ListInt w; ListInt z; z = ListInt(11,&w);
ListInt t; t = z;
ListInt* tt=new ListInt(11,new ListInt(22,new ListInt()));
ListInt ttt = *tt; ListInt* p = tt;
```

Ajoutez à la classe **ListInt** les méthodes suivantes :

- **int longueur()** : retourne la longuer de la liste.
- **void addVal(int pos, int val)** : ajoute la valeur **val** en position **pos** dans une liste existante.
- **ListInt* addVal(int pos, int val)** : crée une nouvelle liste, ne partageant aucune mémoire avec des listes existantes, obtenue en ajoutant la valeur **val** en position **pos** dans la liste courante.
- **int getVal(int pos)** : retourne la valeur située en position **pos** dans une liste.

Ces méthodes doivent être simples, concises, et récursives.