Nous rappelons que les méthodes sont maintenant à écrire dans la classe correspondante.

Sauf mention contraire, il est interdit d'utiliser des méthodes des exercices précédents pour résoudre l'exercice courant!

Les boucles sont interdites.

1 Rappels

On considère le squelette de classe liste suivant.

```
class Liste{
    private int val;
    private Liste suiv;

    public Liste(){
      suiv = null;
    }

    public boolean estVide(){
        return suiv==null;
    }
}
```

1.1 Exercices où l'on ne modifie pas this

Exercice 1. Longueur

Question 1.1.
Ecrire une méthode `int longueur()` qui calcule la longueur de la liste courante.

Exercice 2. Recherche

Question 2.1.
Ecrire une méthode `boolean recherche(int x)` qui retourne vrai ssi x est dans la liste courante.

Exercice 3. Croissant

Question 3.1.
Ecrire une méthode `boolean croissant()` qui retourne vrai ssi les entiers de la liste courante sont triés par ordre croissant.

1.2 Exercices où l'on modifie this

Dans cette partie, vous pouvez utiliser les méthodes

```
public void ajoutTete(int x){
    Liste aux = new Liste();
    aux.val = val;
    aux.suiv = suiv;
    this.val = x;
```

```
        this.suiv = aux;
    }

    public void supprimeTete(){
        //sur liste non vide
        this.val = suiv.val;
        this.suiv = this.suiv.suiv;
    }
```

On rappelle que "this = " est interdit en Java.

Exercice 4. AjoutFin

Question 4.1.
Ecrire une méthode `void ajoutFin(int x)` qui modifie la liste courante en ajoutant x à la fin.

Exercice 5. Concaténation

Question 5.1.
Ecrire une méthode `void concat(Liste l)`. Pré-requis : l n'a aucun maillon en commun avec la liste courante. Action (décrite informellement) : raccorde la fin de this avec le début de l.

Question 5.2.
Que voit on affiché dans le programme suivant ?

- Liste L1 = (1); Liste L2 = (2,3);
- L1.concat(L2);
- System.out.println(L1);
- L2.val = 50;
- System.out.println(L1);
- L2.suiv.val = 51;
- System.out.println(L1);

Exercices bonus

Exercice 6. Copie

Question 6.1.
Ecrire une méthode `Liste copie()` qui construit une copie indépendante de la liste courante.

Exercice 7. Get

Question 7.1.
Ecrire une méthode `int get(int i)` qui pour tout i, $0 \leq i < this.longueur()$, renvoie l'entier en position i de la liste courante.

Exercice 8. AjoutFin

Question 8.1.
Ecrire une méthode `Liste ajoutFin(int x)` qui renvoie une nouvelle liste indépendante de la liste courante, avec la valeur x rajoutée à la fin.

Exercice 9. ListeTriee On considère la classe ListeTriee définie de la même manière que la classe Liste, mais en ajoutant l'invariant que les entiers de la liste sont triés par ordre croissant.

Question 9.1.
Ecrire une méthode `ListeTriee fusion(ListeTriee l)` qui renvoie une nouvelle ListeTriee indépendante de la liste courante et de l, et correspondant à la fusion de la liste courante et de l. (Vous pouvez utiliser la méthode Liste copie())

Question 9.2.
Refaire la question précédente SANS utiliser la méthode Liste copie()

Question 9.3.
Ecrire une méthode `void insereDansTriee(int x)` qui modifie la liste courante (supposée triée par ordre croissant) en insérant x au bon endroit.