Projet 1 : Ensembles (3 séances, à rendre)

1. Introduction

Ce projet consiste à comparer plusieurs implémentations d'ensembles. Les ensembles à implémenter disposent des méthodes décrites dans le sujet du premier TP auxquelles s'ajoutent les méthodes d'intersection, d'union et de différence d'ensembles.

2. Implémentations

a) Chaînage simple.

La première implémentation à fournir est celle du premier TP : chaînage simple sans doublon, non trié, ajout en queue de chaînage, nombre de maillons mémorisé.

b) Tableau trié.

Écrivez une seconde implémentation utilisant un tableau statique, ajout en ordre (le type T est supposé répondre à l'opérateur « < »).

c) Chaînage simple avec doublon.

Écrivez une troisième implémentation utilisant un chaînage simple non trié, mais ne testant pas la présence d'un élément lors de l'ajout (un même élément peut apparaître plusieurs fois dans le chaînage). Bien sûr la méthode `retire` doit être adaptée pour que l'élément à supprimer ne figure plus du tout après. L'ajout se fait cette fois en tête (pourquoi ? Discuter dans le rapport).

3. Mise en place

Les différentes implémentations consistent à écrire la même classe (Ensemble) mais sous des noms légèrement différents dans plusieurs fichiers. Par exemple `Ensa, Ensb, Ensc` pour faire référence aux questions de l'énoncé. Par contre, les méthodes doivent avoir la même signature.

Un programme de test, écrit au fur et à mesure du développement, doit permettre de vérifier le bon fonctionnement de chaque méthode (tests unitaires). Un même code doit être appliqué à chaque implémentation pour s'assurer de leur interchangeabilité. La version finale de ce programme doit figurer dans l'archive rendue.

4. Rendu

Un document de travail doit être rédigé **au préalable et en parallèle**, précisant les ambiguïtés du sujet. Ce document présentera aussi les choix effectués (constantes, comportement des méthodes non précisé dans l'énoncé, …) et les limites d'utilisation. Il est aussi, si ce n'est plus important, que le code. Il devra comporter l'ordre de grandeur du coût de chaque méthode. Les implémentations doivent être discutées : en particulier quelle est la taille des données, la forme des données au mieux et au pire pour chaque méthode, et les ordre de grandeur associés (justifiés en quelques mots). Des modifications peuvent être proposées et discutées pour accélérer le traitement ou diminuer l'empreinte mémoire, mais sans être implémentées.

Le travail est à déposer sur MADOC au plus tard le 12 mars 2017 à 23h55 sous forme d'une archive zip contenant le rapport (pdf) et le code (plusieurs fichiers texte).

Le code doit être clair et concis, commenté, avec en particulier les pré- et post-conditions. Le rapport doit être bien écrit (français correct, attention à l'orthographe !) et agréable à lire, il doit comporter introduction et conclusion et des annexes pour les détails techniques (dessins, code, etc.). Un tableau similaire au tableau ci-dessous doit récapituler les coûts temporels au pire obtenus pour chaque méthode et chaque implémentation. Vous pouvez ajouter des colonnes pour des implémentations que vous n'avez pas codées mais dont vous imaginez les coûts. Précisez dans ce cas en quoi elles se distinguent de celles demandées (tri, chaînage double, ajout en tête,…) et si leurs coûts spatiaux différent.

	a) chaînage simple sans doublon non trié ajout en queue	b) tableau trié sans doublon	c) chaînage simple avec doublons non trié ajout en tête	
`Ensemble`	`O(1)`			
`~Ensemble`				
`estVide`		`O(1)`		
`contient`				
`ajoute`				
`retire`				
`contenu`				
`inter`				
`union`				
`moins`				