TP Collection de Points

Nous souhaitons écrire une classe qui permet de stocker et de gérer un ensemble de Points. Les attributs de la classe sont : un pointeur sur Point ***(Point *)*** qui sera le tableau de stockage ; un nombre de Points ***nbp*** qui doit être initialisé à 0 à la création ; une capacité ***(cap)*** (désigne la taille du tableau qui sera alloué dynamiquement). Utiliser la classe point du TP précédent

0- Ajouter à la classe Point une fonction qui dit si deux points sont identiques : deux points sont identiques si la distance les séparant est inférieur à 0.0001.

1- Écrire le constructeur (avec une capacité par défaut = 100)

2- Écrire le constructeur qui initialise la collection en utilisant un tableau de ***Point***s.

3- Écrire le destructeur

4- Écrire la fonction ***present()*** qui renvoie vrai si l'élément donné en argument est présent dans la collection.

5- Écrire la fonction ajouter qui ajoute un ***Point*** s'il n'est pas déjà dans la collection. Si l'ajout ne se fait pas la fonction renvoie ***false***.

6- Écrire la fonction ***supprimer()*** qui supprime de la collection un ***Point*** donné en argument. Si le ***Point*** en question n'existe pas, alors la fonction doit renvoyer ***false.***

7- Écrire la fonction ***ajouter_bis(Point & P)*** qui lorsque la collection est pleine, alloue un espace 2 fois plus grand, y mettre le contenu actuel de la collection et ajoute le ***Point*** P. N'oubliez pas de libérer l'espace inutile désormais.

8- Écrire une fonction membre qui ajoute (utiliser la fonction ajouter_bis) les éléments d'un tableau de ***Point***s à une collection de ***Point***s.

9- On appelle constructeur par recopie, le constructeur ayant le prototype suivant :

col_Points::col_Points(const col_Points &) ;

Ce constructeur nous permettra ainsi d'écrire :

col_Points B(A) ;

ce qui permet de créer l'objet B comme une copie de l'objet A. Par contre les deux objets ne doivent avoir espace mémoire commun. Attention surtout ne pas faire : ***(*this).T=A.T*** !!!!!!!!!!!!!!!!!!!!!!!!!!! (le tableau est partagé, ce que ne l'on souhaite pas). Écrire le constructeur par recopie.

10- Écrire la fonction :

void col_Points::intersection(const col_Points & A, col_Points & B).

qui met l'intersection de ***(*this)*** et de A dans B.

11 - Écrire un constructeur qui prend 2 collections de points en argument et qui crée la collection union des 2 autres.

12- Écrire une fonction qui renvoie 5 résultats : le point ayant la plus petite abscisse, le point ayant la plus grande abscisse, le point ayant la plus petite ordonnée, le point ayant la plus grande ordonnée et le centre de tous les points de la collection.

Toutes les fonctions doivent être testées avec des affichage adéquats, explicites et clairs

Le programme doit être organisé en plusieurs fichiers .cpp et .h et accompagné d'un fichier makefile. L'exécutable doit pouvoir être fabriqué grâce à la commande make.