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

Votre programme doit contenir : ***Point.h, Point.cpp, col_Points.h, col_Points.cpp, principal.cpp***
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.

Toutes les fonctions doivent être testées avec des affichage adéquats, explicites et clairs. Commencez à tester dès que possible. Une fonction non testée est considérée comme non faite.

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 destructeur

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

4- Écrire la fonction ***ajouter(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. Attention, la collection ne doit pas contenir de doublon.

5- Écrire la fonction d'affichage

6- Écrire le constructeur par recopie

7- Écrire la fonction :

void col_Points::union(const col_Points & A).

qui ajoute tous les éléments de A à la collection (*this). Attention, la collection ne doit pas contenir de doublons.

8- Écrire une fonction qui renvoie 2 résultats : le centre et le point le plus loin du centre

9- Dans le main créer une collection vide à la quelle vous ajouter (grâce à une boucle) 100 points initialisé aléatoirement. Afficher le temps nécessaire à la recherche d'un point donné (quand le point n'existe pas dans la collection).

Faites la même chose pour 1000, 10000, 1000000.