TP3 / C++

Exercice 1 :

Soit la classe *Point3D* définie comme suit (dans *Point3D.hpp*) :

```
class Point3D {
	private:
		float x,y,z;	// private attributes

	public:
		// constuctors
		Point3D(); // fill X Y Z with random values (from 0 to 100)
		Point3D(const float &newx, const float &newy, const float &newz);
// fill XYZ values

		// Setters and getters
		void setXYZ(const float &newx, const float &newy, const float
&newz);
		void setX(const float &newx);
		void setY(const float &newy);
		void setZ(const float &newz);
		float getX();
		float getY();
		float getZ();

		// other methods
		void print();	// prints the coordinates of the point
		float distanceTo(const Point3D &otherPoint3D);
};
```

1) Développer l'ensemble des méthodes dans le fichier *Point3D.cpp*.

2) Tester les différentes méthodes dans la fonction *main()*.

Remarque : Pour la génération de nombres aléatoires, faire appel aux fonctions *rand()* et *srand()* de la librairie *<cstdlib>*.

Exercice 2 :

Soit la classe *Trajectory* définie comme suit (dans *Trajectory.hpp*) :

```
#include "Point3D.hpp"

constexpr size_t numberOfPoints = 10;
class Trajectory{
	private:
		Point3D points[numberOfPoints];
	public:
		void print(); // print the coordinates of all points
		Point3D & getPoint(const int &n); // returns the reference of point n
		float getTotalDistance();
};
```

1) Développer l'ensemble des méthodes dans le fichier *Trajectory.cpp*.

2) Tester les différentes méthodes dans la fonction *main()*.

3) Modifier l'ensemble des points pour qu'ils soient sur une droite. Comparer la distance totale avec la distance entre le premier et le dernier point.

3) Modifier la classe pour que le nombre de points soit dynamique. Le constructeur prend obligatoirement le nombre de points en paramètre et alloue l'espace nécessaire. Il faut aussi développer un destructeur pour la classe.