TP 3 : Programmation C++ correction

Exercice 1 :

```
#include<iostream>

using namespace std;

class Etudiant
{
    int matricule;
    string nom;
    int nbrNotes;
    float* tabNotes;
    static int n;

public:

    Etudiant()
    {
        matricule = 0;
        nom = "";
        nbrNotes = 0;
        tabNotes = new float[nbrNotes];
    }

    Etudiant(string nome, int nb)
    {
        n++;
        matricule = n;
        nom = nome;
        nbrNotes = nb;
        tabNotes = new float[nbrNotes];
    }

    ~Etudiant()
    {
        delete[]tabNotes;
    }

    Etudiant(const Etudiant& E)
    {
        matricule = E.matricule;
        nom = E.nom;
        nbrNotes = E.nbrNotes;
        tabNotes = new float[nbrNotes];
        for (int i = 0; i < nbrNotes; i++)
        {
            tabNotes[i] = E.tabNotes[i];
        }
    }

    int getMatricule()
    {
        return matricule;
```

```
    }

    string getNom()
    {
        return nom;
    }

    int getNbNotes()
    {
        return nbrNotes;
    }

    void setNom(const string& n)
    {
        nom = n;
    }

    void saisie()
    {
        cout << "Saisie des notes : " << endl;
        for (int i = 0; i < nbrNotes; i++)
        {
            cout << "Donner la note " << i + 1 << " : ";
            cin >> tabNotes[i];
        }
    }

    void affichage()
    {
        cout << "- Matricule : " << matricule << endl
            << "- Nom : " << nom << endl
            << "- Nombre de notes : " << nbrNotes << endl
            << "- Notes : " << endl;
        for (int i = 0; i < nbrNotes; i++)
        {
            cout << tabNotes[i] << "\t";
        }
        cout << endl;
    }

    float moyenne()
    {
        float s = 0;
        for (int i = 0; i < nbrNotes; i++)
            s = s + tabNotes[i];

        return s / nbrNotes;
    }

    bool admis()
    {
        return (moyenne() >= 10);
    }

    bool comparer(Etudiant E)
    {
        return (moyenne() == E.moyenne());
    }
    float get_tab();
};
float Etudiant ::get_tab()
{
    cout << "get : " << endl;
```

```
    for (int i = 0; i < nbrNotes; i++)
    {
       return tabNotes[i];
    }
}

//Intialisation du membre statique
int Etudiant::n = 0;

int main()
{
    Etudiant E("etud1", 3);
    cout << "Creation d'un objet Etudiant E avec 3 notes" << endl;
    E.saisie();
    cout << "Affichage de l'etudiant E:" << endl;
    E.affichage();
    cout << endl;
    E.get_tab();

    cout << "Copie de l'etudiant E dans E1" << endl;
    Etudiant E1(E);
    cout << "L'affichage de l'etudiant E1:" << endl;
    E1.affichage();
    cout << endl;
    Etudiant E2("etud2", 2);
    cout << "Creation d'un objet Etudiant E2 avec 2 notes" << endl;
    E2.saisie();
    cout << endl << "Appel des getters :" << endl;
    cout << "Le matricule de l'etudiant E2 : " << E2.getMatricule() << endl;
    cout << "Le nom de l'etudiant E2 : " << E2.getNom() << endl;
    cout << "Le nombre de notes de l'etudiant E2 : " << E2.getNbNotes() << endl;

    cout << endl << "La moyenne de l'etudiant E est : " << E.moyenne() << endl;

    if (E.admis())
    {
        cout << "l'etudiant E est admis " << endl;
    }
    else
    {
        cout << "l'etudiant E n'est pas admis " << endl;
    }

    if (E.comparer(E2))
    {
        cout << "Les Etudiants E et E2 ont la meme moyenne " << endl;
    }
    else
    {
        cout << "Les Etudiants E et E2 n'ont pas la meme moyenne " << endl;
    }

    return 0;
}
```

Exercice 2 :

```
#include<iostream>
#include<cmath>
using namespace std;
class vecteur3d {
      float x;
```

```
	float y;
	float z;

public:

	//Constructeur d'initialisation
	vecteur3d(float a = 0, float b = 0, float c = 0) : x(a), y(b), z(c) {
	}

	//Constructeur de recopie
	vecteur3d(const vecteur3d& v) {
		x = v.x;
		y = v.y;
		z = v.z;
	}

	//L'affichage d'un vecteur
	void afficher() {
		cout << "(" << x << "," << y << "," << z << ")" << endl;
	}

	//La somme de deux vecteur
	vecteur3d somme(const vecteur3d& v) {
		vecteur3d s;
		s.x = x + v.x;
		s.y = y + v.y;
		s.z = z + v.z;
		return s;
		//Ou return vecteur3d(x+v.x, y+v.y, z+v.z);
	}

	//Le produit scalaire de deux vecteurs
	float produit(const vecteur3d& v) {
		return x * v.x + y * v.y + z * v.z;
	}

	//tester si deux vecteurs ont les memes composantes
	bool coincide(const vecteur3d& v) {
		return (x == v.x && y == v.y && z == v.z);
	}

	//Retourner la norme du vecteur
	float norme() {
		return sqrt(x * x + y * y + z * z);
	}

	//Retourner le vecteur qui la plus grande norme : par valeur
	vecteur3d normax(vecteur3d v) {
		if (this->norme() > v.norme())
			return *this;

		return v;
	}

	//Retourner le vecteur qui la plus grande norme : par adresse
	vecteur3d* normax(vecteur3d* v) {
		if (this->norme() > v->norme())
			return this;

		return v;
	}

	//Retourner le vecteur qui la plus grande norme : par reference
```

```
	vecteur3d& normaxR(vecteur3d& v) {
		if (this->norme() > v.norme())
			return *this;

		return v;
	}
};

int main() {
	vecteur3d v1(1, 2, 3);
	cout << "Vecteur V1";
	v1.afficher();
	vecteur3d v2(5, 6, 7);
	cout << "Vecteur V2";
	v2.afficher();
	cout << endl;
	cout << "La somme des vecteurs v1 et v2 est : ";
	(v1.somme(v2)).afficher();
	cout << "Le produit scalaire des vecteurs v1 et v2 est : " <<
v1.produit(v2) << endl;
	cout << endl;
	cout << "Copier le vecteur V1 dans V3:" << endl;
	vecteur3d v3(v1);
	cout << "Vecteur V3";
	v3.afficher();
	if (v1.coincide(v3))
		cout << "Les vecteurs v1 et v3 coincident " << endl;
	else
		cout << "Les vecteurs v1 et v3 ne coincident pas " << endl;

	cout << endl;
	cout << "Le vecteur qui a la plus grande norme est (par valeur): ";
	(v1.normax(v2)).afficher();
	cout << "Le vecteur qui a la plus grande norme est (par adresse): ";
	(v1.normax(&v2))->afficher();
	cout << "Le vecteur qui a la plus grande norme est (par reference) :";
	(v1.normaxR(v2)).afficher();
	cout << endl;
}
```