Atelier 3 Khadija Diaz Grp 2

Exercice 1:

```
#include <iostream>

using namespace std;
class A{
public:
void display()
{
cout<<"La methode display de la classe A est execute."<<endl;
}
};
class B:public A{
public:
void display()
{
cout<<"La methode display de la classe B est execute."<<endl;
};

};
int main()
```

```
{
B b;
b.display();
return 0;
}
```

Exercice 2:

```
#include <iostream>
using namespace std;
class Shape
{
protected:
  float x, y;
public:
  Shape(float a, float b)
  {
    x = a;
    y = b;
  }
};
class Rectangle: public Shape
{
public:
```

```
  Rectangle(float a, float b) : Shape(a, b) {}
  float area()
  {
    return (x * y);
  }
};
class Triangle: public Shape
{
public:
  Triangle(float a, float b) : Shape(a, b) {}
  float area()
  {
    return (x * y / 2);
  }
};
int main (){
  Rectangle rectangle(2,3);
  Triangle triangle(2,3);
  cout << rectangle.area() << endl;
  cout << triangle.area() << endl;
  return 0;
}
```

Exercice 3:

```
#include <iostream>
using namespace std;
class Complexe{
public:
   Complexe(float x,float y);
   Complexe();
   void lis();
   void affiche();
   Complexe operator+(Complexe g);
private:
   float re,im;
};
Complexe::Complexe(float x ,float y)
{
   re=x;
   im=y;
}
Complexe::Complexe(){
```

```
    re=0;
    im=0;
}
void Complexe::lis(){
    cout<<"Partie reel ?"<<endl;
    cin>>re;
    cout<<"Partie imaginaire ?"<<endl;
    cin>>im;
}
void Complexe::affiche(){
    cout<<re<<"+i"<<im;
}
 Complexe Complexe::operator+(Complexe g){
    return Complexe(re+g.re,im+g.im);
}
int main()
{
Complexe Z1(0,1);
Complexe Z2;
Z1.affiche();
cout<<endl<<"entrez un nombre complexe"<<endl;
Z2.affiche();
```

```
cout<<"Vous avez entre:";
Z2.affiche();
Complexe Z3=Z1+Z2;
cout<<endl<<"le somme de" <<endl;
Z1.affiche();
cout<<endl<<"et"<<endl;
Z2.affiche();
cout<<endl<<"est"<<endl;
Z3.affiche();
return 0;

}
```

Exercice 4:

```
#include <iostream>

using namespace std;
class MyClass
{
  public:
    MyClass()
      {
```

```
     cout<<"Je suis le constrectur"<<endl;
    }
    ~MyClass()
    {
       cout<<"je suis le destrectur."<<endl;
    }
};
int main()
{
  MyClass v1;
   return 0;
}
```

Exercice 5:

```
#include <iostream>
#include<cstring>
using namespace std;
class Animal{
protected:
   int age;
  string name;
```

```
public:
void set_value(int a,string b)
{
   age=a;
   name=b;
    //strcpy(b,name);
}
};
class Zebra:public Animal{
public:
void messagezebra()
{
   cout<<"The Zebra named "<<name<<" is "<< age <<" year old. The Zebra comes from Africa"<<endl;
}

};
class Dolphin:public Animal{
public:
void messagedolphin()
{
```

```
   cout<<"The Dolphin named "<< name <<" is "<< age <<" year old. The Dolphin comes from New Zeland"<<endl;
}


};

int main()
{
  Zebra zebr;
  Dolphin dol;
  zebr.set_value(5,"Nana");
  dol.set_value(8,"Doli");
  zebr.messagezebra();
  dol.messagedolphin();
   return 0;
}
```

Exercice 6:

```
#include <iostream>

using namespace std;
   class Personne
   {
   private :
      string nom;
       string prenom;
       double datenaissance;

   public:
      Personne(string n, string p, double dn)
      {
         nom = n;
         prenom = p;
         datenaissance = dn;
      }

       virtual void Afficher()
      {
         cout<<"Nom: " << nom << " Prénom: " << prenom << " Date de
naissance: " << datenaissance<<endl;
```

```
    }
};
class Employe : Personne
{
private:
    double salaire;

public:
    Employe(string n, string p, double dn, double s) : Personne(n, p, dn)
    {
        salaire = s;
    }

    void Afficher()
    {
        cout<<" Salaire: " << salaire;
}
};

class Chef : Employe
{
private:
```

```
    string service;
    public:
        Chef(string n, string p, double dn, double s, string ser): Employe(n, p,
dn, s)
    {
      service = ser;
    }

    void Afficher()
    {

      cout<<" Service: " << service;
    }
  };
class Directeur : Chef
  {
    private:
      string societe;

    public :
    Directeur(string n, string p, double dn, double s, string ser, string
soc):Chef(n, p, dn, s, ser)
```

```
    {
        societe = soc;
    }

     void Afficher()
    {

        cout<<" Société: " << societe;
    }
};

     int main()
    {
    return 0;
}
```

Exercice 7:

```
#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;
    return 0;
}
```

Exercice 8: