UNIVERSITÉ DE MONTPELLIER

FACULTÉ DES SCIENCES MONTPELLIER

M2 Modélisation et Analyse Numérique
UE Programmation 2 - 2022/2023

CC 3 - Inheritance

les fichiers à rendre à l'issue de cette évaluation sont indiqués en rouge, alors que les fichiers fournis sont indiqués en bleu (dans le dossier /CC2). merci de respecter la dénomination demandée.

We consider the ComplexNumber class designed in PW2 and realize that any ComplexNumber instance may also be regarded as a point in the $(\vec{x}, \vec{y})$ plane. This clearly enters the "is a" relationship that grounds the inheritance mechanism. A class point is provided as two files class_Point2D.hpp and class_Point2D.cpp. Here is the declaration of this class:

```
class point{
  double x;
  double y;

public:
  point(double xx=0.,double yy=0.):x(xx),y(yy){};   //  arguments with default values
  inline double X() const{return x;};
  inline double Y() const{return y;};
  inline void zero(){x=y=0.;};
  ~point(){};   //  destructor
  point(const point& p):x(p.x),y(p.y){}   //  copy constructor
  const point& operator=(const point& p); //  point-to-point assignment
  const point& operator+=(const point& p); //  adding a point to the current point
  const point operator+(const point& p) const; //  add two points
};
```

The goal of the evaluation is to re-design the ComplexNumber **class so that it publicly derives from the** point **class**, with the following public members:

(1) the first constructor of ComplexNumber, with the following prototype:

```
complex(double xx, double yy)
```

should explicitly make use of the point constructor through the initialization list,

(2) a second constructor that accepts a point argument as input:

```
complex(const point&p);
```

(3) an overloaded version of complex::**operator**=(), with the following prototype:

```
const complex& operator=(complex& z);
```

(4) an overloaded version of complex::**operator**+=(), with the following prototype:

```
const complex& operator+=(complex& z);
```

which relies on the the point::**operator**+=(),

(5) your class should compile and link against the following main() program:

```
#include <iostream>
#include "class_Complex.hpp"

using namespace std;
```

```
int main()
{
  complex b(1., 1.);
  complex c(2., 2.);
  complex d(3., 3.);

  d = c;
  cout << "d.x= " << d.X() << endl;
  cout << "d.y= " << d.Y() << endl;

  d += c;
  cout << "d.x= " << d.X() << endl;
  cout << "d.y= " << d.Y() << endl;
}
```

The resulting class should be provided as a pair of files class_Complex.hpp and class_Complex.cpp.

Fin du sujet.