UEE 1303(1067): Object-Oriented Programming

Lab #8: Inheritance (I)

In this laboratory session you will:

- learn the concept of inheritance and its usage.

Lab 8-1: Example of Inheritance

✓ Please compiler and execute the program `lab8-1-1`, where `Point4D` is a derived class from the base class `Point2D`.

```
// lab8-1-1.cpp
#include <iostream>
using std::cout; using std::endl;
class Point2D
{
public:
    Point2D(int n1 = 0, int n2 = 0):x(n1), y(n2){}
    void display() const;
private:
    int x;
    int y;
};
void Point2D::display() const
{
    cout << x << "," << y;
}
class Point4D : public Point2D
{
public:
    Point4D(int n1 = 0,int n2 = 0,int n3 = 0,int n4 =
0):Point2D(n1,n2),z(n3), t(n4){}
    void display() const;
private:
    int z;
    int t;
};
```

```
void Point4D::display() const
{
    Point2D::display();
    cout << "," << z << "," << t;
}
int main()
{
    Point4D pt(1,2,3,4);
    pt.display();
    return 0;
}
```

- Note that `Point4D` has member of class `Point2D` in addition to its own members.
- You can put the constructor of the base class in the initialization list for the derived class.
- Note that member function of a derived class cannot access the private part of a base class. For example, the function `Point4D::display()` cannot be defined as

  ```
  void Point4D::display() const
  {
      cout << x << "," << y; // x and y are inaccessible
      cout << "," << z << "," << t;
  }
  ```

  The hidden member `x` and `y` of the derived class `Point4D` is accessible through the public member function `Point2D::display();`.
- You can define *accessor* and *mutator* functions in `Point2D` to access private members.

✓ Please compiler and execute the program `lab8-1-2`

```
// lab8-1-2.cpp
/* The Point2D and Point4D class defined in lab8-1-1 */
int main()
{
    Point2D pt2(3,4);
    Point4D pt4(1,2,3,4);
    pt4.display(); cout << endl;
    pt2 = pt4; // OK, every Point2D is a Point4D
```

```
    pt2.display(); cout << endl;
    pt4 = pt2; // Error, not every Ponint4D is a Point2D
    pt4.display(); cout << endl;
    return 0;
}
```

- You can comment the incorrect lines to observe the results.
- If you require type conversion from a base class to derived class (eg. `pt4 = pt2`), you have to provide additional member functions of `Point4D` to achieve it.

✓ Please compiler and execute the program `lab8-1-3`

```
// lab8-1-3.cpp
/* The Point2D and Point4D class defined in lab8-1-1 */
void f(const Point2D &p1, const Point2D &p2)
{
    p1.display(); cout << endl;
    p2.display(); cout << endl;
}
int main()
{
    Point2D pt2(3,4);
    Point4D pt4(1,2,3,4);
    f(pt2,pt4);
    return 0;
}
```

- Note that the prototype of function f is `void f(const Point2D &, const Point2D &)`.

Lab 8-2: Class Hierarchy

✓ A derived class can be a base class of another derived class.

```
// lab8-2.cpp
/* The Point2D and Point4D class defined in lab8-1-1 */
class Car : public Point4D
{
public:
    Car(int n1 =0, int n2 = 0, int n3 = 0,
                   int n4 = 0):Point4D(n1,n2,n3,n4)
```

```
    {
        color = 0;
        year = 0;
    }
    // copy constructor
    Car(const Point4D &p):Point4D(p){color = 0; year = 0;}
    void display() const;
    void setColor(const int c){color = c;}
    void setYear(const int y){year = y;}
private:
    int color;
    int year;
};
void Car::display() const
{
    cout << "color: " << color << endl;
    cout << "year: " << year << endl;
    Point4D::display();
}
int main()
{
    Point4D pt4(1,2,3,4);
    Car c1(pt4);
    c1.setColor(128);
    c1.setYear(2011);
    c1.display(); cout << endl;
    return 0;
}
```

- Note that, to enable copy constructor of `Car`, you should also provide copy constructor for `Point2D` and `Point4D`.

Exercise 8-1

✓ Please modify the class `Point2D` and `Point4D` defined in `lab8-1`. In `Point2D`, the member `x` and `y` become two pointers to integer, respectively. Similarly, the member `z` and `t` should be changed as pointers. The modified classes are shown as follows,

```
// Point2D.h
#ifndef POINT2D_H
#define POINT2D_H
class Point2D
{
public:

private:
    int *x;
    int *y;
};
#endif
```

✓ Please implement `Point2D` and `Point4D` in different files.

```
// Point4D.h
#ifndef POINT4D_H
#define POINT4D_H
class Point4D
{
public:

private:
    int *z;
    int *t;
};
#endif
```

✓ Please finish the remaining part to make the following main function work successfully.

```
#include <iostream>
#include "Point2D.h"
#include "Point4D.h"
using std::cout; using std::endl;
int main()
{
    Point2D pt1(1,2);
    Point2D pt2(3,4);
```

```
    pt1.display(); cout << endl;
    pt2.display(); cout << endl;
    pt2 = pt1;
    pt2.display(); cout << endl;
    Point4D pt4(5,6,7,8);
    pt4.display(); cout << endl;
    pt2 = pt4;
    pt2.display(); cout << endl;
    pt4 = pt1;
    //pt4 could be (1,2,7,8) or (1,2,0,0)
    pt4.display(); cout << endl;
    return 0;
}
```