Exercise 05 - OOP: Abstraction & Encapsulation

Create a header file for each task such that its name is the same as the task's class.

1. Define class *Square* that contains
   - a private double field named *length*.
   - a public default constructor that assigns 1 to its field.
   - a public copy constructor.
   - a public overloaded assignment operator.
   - a public empty destructor.
   - a public double constant method named `side()` that takes no parameters and returns *length*.
   - a public void method named `side()` that takes a double parameter. It assigns the parameter to *length* only if the parameter is positive.
   - a public string constant method named `toString()` that takes no parameters and returns a string in the format

     `[`*x*`]`

     where *x* is the value of *length* with two decimal places.
   - a friend overloaded ostream operator that displays its output in the same format as `toString()`.

2. Define class *Rectangle* that contains
   - a private double field named *lth*.
   - a private double field named *wth*.
   - a public default constructor that assigns 1 to each field.
   - a public copy constructor.
   - a public overloaded assignment operator.
   - a public empty destructor.
   - a public double constant method named `length()` that takes no parameters and returns *lth*.
   - a public double constant method named `width()` that takes no parameters and returns *wth*.
   - a public void method named `length()` that takes a double parameter. It assigns the parameter to *lth* only if the parameter is positive and greater than or equal to *wth*.
   - a public void method named `width()` that takes a double parameter. It assigns the parameter to *wth* only if the parameter is positive and less than or equal to *lth*.
   - a public string constant method named `toString()` that takes no parameters and returns a string in the format

     `[`*x*`,`*y*`]`

     where *x* and *y* are the values of *wth* and *lth* respectively with two decimal places.
   - a friend overloaded ostream operator that displays its output in the same format as `toString()`.

3. Define class *FourTuple* that contains
   - a private int array field named *values* with a size of 4.
   - a public default constructor that assigns 0 to each element of *values*.
   - a public copy constructor.
   - a public overloaded assignment operator.
   - a public empty destructor.
   - a public overloaded subscript operator with an integer reference return value that takes an integer parameter. If the parameter is between 0 and 3 inclusively, it returns the element whose index equals the parameter.

- a public string constant method named `toString()` that takes no parameters and returns a string in the format

  $$(w,x,y,z)$$

  where w, x, y and z are the values of elements of *values* from the first to the last.
- a friend overloaded ostream operator that displays its output in the same format as `toString()`.

4. Define class *Pin* that contains

   - a private string field named *pin*.
   - a private Boolean field named *view*.
   - a public default constructor that assigns `"1234"` and false to *pin* and *view*, respectively.
   - a public copy constructor.
   - a public overloaded assignment operator.
   - a public empty destructor.
   - a public string constant method named `passcode()` that takes no parameters and returns *pin*.
   - a public void method named `passcode()` that takes a string parameter. It assigns the parameter to *pin* only if the parameter is a string of 4 digits.
   - a public void method named `show()` that takes a Boolean parameter and assigns the parameter to *view*.
   - a public string constant method named `toString()` that takes no parameters and returns a string in the format

     $$\begin{cases} x & \text{if } view \text{ is true} \\ \texttt{"****"} & \text{if } view \text{ is false} \end{cases}$$

     where x is the value of *pin*.
   - a friend overloaded ostream operator that displays its output in the same format as `toString()`.