Computer Lab #4.

C++ as an Object-Oriented Language.
Classes and Objects

Computer programs usually model aspects of some real or abstract world (the domain). Because each class models a concept, classes provide a more natural way to create such models. Each class in the model represents a noun in the domain, and the methods of the class represent verbs that may apply to that noun. Classes allow a clear correspondence (mapping) between the model and the domain, making it easier to design, build, modify and understand these models. Classes provide some control over the often challenging complexity of such models.

Classes can accelerate development by reducing redundant program code, testing and bug fixing. If a class has been thoroughly tested and is known to be a 'solid work', it is usually true that using or extending the well-tested class will reduce the number of bugs - as compared to the use of freshly-developed or ad hoc code - in the final output. In addition, efficient class reuse means that many bugs need to be fixed in only one place when problems are discovered.

Task 1.

Develop and implement a *Rectangle* class. Consider the following mandatory functions and restrictions:

a) Any rectangle can be defined by its sides. The sides should not be accessible from the outside of *Rectangle* class.

b) At initialization phase, it should be possible to define the rectangle by:
   - two sides: we get a default rectangle with non-equal sides;
   - one side: we get a square (all sides are equal); square is just a special rectangle.
   - no values, or empty parameter list: in this case the sides are zeroes.

c) Insure the consistency of initial values; the class attributes should always represent a valid rectangle. Users can enter any values (even negative), and it is your task to check the input consistency.

d) The class should provide the following information about the rectangle:
   - length of all sides one after another;
   - its perimeter;
   - its area.

e) Implement *print()* public class operation that prints all information about the rectangle: its sides, area and perimeter values.

Write a *main()* function that contains all necessary testing code to check the correctness of the *Rectangle* class implementation. Any kind of user input should take place only in *main()*, not in the *Rectangle* class. However, validity check of input data is in the responsibility of *Rectangle* class.

Decomposition of the program:

- *rectangle.h* file: class declaration header;
- *rectangle.cpp*: implementation of methods declared in the header;
- *main.cpp*: the main program with testing code.

Task 2.

Based on the example above (see Task 1) develop and implement a *Circle* class. A circle can be defined by its radius *R*, which should be stored as a class attribute. Make sure that a radius is inaccessible from the outside of *Circle* class. Insure the consistency of objects: it should not be possible to bring any circle into inconsistent state. The *Circle* class should provide information about the radius, area and circumference (length) of the circle. Write a testing function (*main*) to test the correctness of your implementation. Follow the program decomposition from Task 1 (header, implementation, main program).