**1.** What are the two parts of a class, as described in the chapter?

Public - it is an interface that specifies how to use the members of class.

Private - it is implementation details of class that are not directly available to the user

**2.** What is the difference between the interface and the implementation in a class?

All facilities of interface are available to the user, while members declared in implementation part can be used only by the other members of class

**3.** What are the limitations and problems of the original Date struct that is created in the chapter?

All members were public by default, so the user could explicitly change the value of object’s member

**4.** Why is a constructor used for the Date type instead of an init\_day() function?

Because it is much more convenient to use constructor to create objects. It also supports invariants as an init\_day() and also we can provide default constructor, which will be used if no arguments were provided to the constructor

**5.** What is an invariant? Give examples.

Invariant is a statement that must be true in the given point of the program. For example a statement in loop (while cin) tells us that the code is executed while input stream is opened

**6.** When should functions be put in the class definition, and when should they be defined outside the class? Why?

If our function consists of one or at most two lines of code, than we can put it in class definition. These functions are called ‘inline functions’. Instead of using the reference to the place where the code of function is, compiler will generate the code of function at each point of call. Thus we can achieve better speed but only when the function has 1 or 2 lines. In all other cases the function should be defined outside the class.

**7.** When should operator overloading be used in a program? Give a list of operators that you might want to overload (each with a reason).

When we want to use conventional operators with user-defined types

!= , == compare two objects of the given type

++ -- increment or decrement the value of the object

<<, >> output or input the value of the object

**8.** Why should the public interface to a class be as small as possible?

a. It has to be easy to understand and use

b. The debugging of small interface is reasonably easier

**9.** What does adding const to a member function do?

It means that the function doesn’t change the value of the given argument and therefore pass-by-const reference can be used

**10.** Why are “helper functions” best placed outside the class definition?

Thus they can’t corrupt the data in the object in case of bug