Exercise 3a. (Container implementation is independent of the itemtype to be stored, 1p)

If your class Time from the exercise 1 does not have the compare function (`operator==`), write and test it first. Overload also the read operator (>>) and implement a constructor with initial values as parameters, e.g. `Time(int hour = 0, int min = 0)`.

At the lectures, we studied the simple list implementation using templates. The application was using the list to store characters. Now we use the same list (available as a list.h file from Oma) to store Time values.

From the Oma-portal you'll find a program main.cpp. This file contains the source code for a small application to test the Time valued list. Test first your Time class with the given test program main.cpp. Then implement a new operation function for the list.h container, `bool insert_to_begin(T item)` which adds a new item to the beginning of the list container. After that change the line `list.insert_to_end(item)` to `list.insert_to_begin(item)` in order to test your new function.

Remark. Use your time as a component. This means that you should include the header file time.h and add time.cpp to the application project.

Extra exercise 3b. (Ordered list, 0.25p)

In this exercise we add an another operation function, `list.insert(item)`, to the list. This operation function insert an item to the list in such a way that the list is always ordered (smallest item first). In this exercise you need to have comparison operator (`operator<`) implemented in your Time class. Verify that the list is always in order.