Class Vector - Task

The Vector container is a wide used data structure. The c-style static array is very inconvenient, since it is a fixed-size compile-time array, and the type of its elements are also set in the compilation process.
We wish to create a dynamic array, which can be created in run-time, can change its size dynamically, and can contain a global type T.

Write (**in C++**) a templated **class Vector** that represents an **array of values** of a general type T. Your code should be flawlessly compiled with the same flags and C++ standard (i.e. C++03) we have used in our training so far.

Provide a full (and tested) implementation for the following **API** to your vector:

Member functions

(constructor)
Construct vector (public member function)
(destructor)
Vector destructor (public member function)
operator=
Assign content (public member function)

Capacity:
size
Return size (public member function)
max_size
Return maximum size (public member function)

resize

Change size (public member function)

capacity

Return size of allocated storage capacity (public member function)

empty

Test whether vector is empty (public member function)

reserve

Request a change in capacity (public member function)

Element access:

operator[]

Access element (public member function)

at

Access element (public member function)

front

Access first element (public member function)

back

Access last element (public member function)

data

Access data (public member function)

Modifiers:

assign

Assign vector content (public member function)

push_back

Add element at the end (public member function)

pop_back

Delete last element (public member function)

insert

Insert elements (public member function)

erase

Erase elements (public member function)

swap

Swap content (public member function)

clear

Clear content (public member function)

You can find the description of each function in the following link:
http://www.cplusplus.com/reference/vector/vector/

Note that you do not have to stick to the exact API demonstrated in the link above.
It is more important for you to understand the required behavior.
More specifically, you are not required to implement an allocator at this stage.

Advanced 1 (Optional) ----------

In addition, implement the following functions:

```
/* Concatenate 2 vectors to a single vector */
operator+
operator+=

/* Output operator */
operator<<
```

Advanced 2 (Optional) ----------

Add to the class a support of iterators. You can read about it on the web in order to get an idea what is needed for that.
class Vector will need to implement the following public methods:

Iterators:

begin

Return iterator to beginning (public member function)

end

Return iterator to end (public member function)