7.2 DIY Vector (graded assignment)

Due 21 Oct 2019 by 23:59 **Points** 10 **Submitting** an external tool

In this final assignment, you are going to implement your own *vector* class. Actually, we are using the following small and simplified subset of the interface from *std::vector* to keep things manageable:

```
template <typename T> class DiyVector{
    public:
        DiyVector();
        ~DiyVector();

        T& at(unsigned int index) const;
        // item at index
        // throws OutOfRange

        unsigned int size() const;
        // number of items in the vector

        void pushBack(const T& item);
        // append item at the end of vector

        void popBack();
        // remove item at the end of vector
        // throws OutOfRange

        void erase(unsigned int index);
        // remove element at index
        // throws OutOfRange

        void insert(unsigned int index, const T& item);
        // insert item before element at index, with:
        // 0 <= index <= size()
        // throws OutOfRange

        class OutOfRange{};

    private:

        // your implementation goes here!
};
```

Your *DiyVector* will store items of type *T* in an array that is dynamically allocated on the heap, using *new[]*. (Compare zyBook Section 12.3, under "Exploring further".)
We only expand this array, but never shrink it:

- when adding an item (using *pushBack* or *insert*), if there is no currently unused element in the array, create a new array, able to hold one more element, and copy over the old items. Of course, you *delete[]* the old, too-short array afterwards.
- when deleting an item (using *popBack* or *erase*), you simply "pack" the array by copying all items behind the erased one to close the gap (and adjust the bookkeeping of used elements). Later, when a new item is to be added (using *pushBack* or *insert*), you re-use an old space at the end of the array, instead of creating a whole new array.

Your program must not use any item storage but arrays that are created by *new[]*. (No statically created arrays, no vectors, no other container classes from std::)
You deliver your *DiyVector* as a header file called *diyvector.h*. It will be automatically tested by the following program, **vector-tester.cpp**. Please download this program for use while you develop your *DiyVector*.

This tool needs to be loaded in a new browser window

Load 7.2 DIY Vector (graded assignment) in a new window