```
/* The following code example is taken from the book
 * "The C++ Standard Library - A Tutorial and Reference, 2nd Edition"
 * by Nicolai M. Josuttis, Addison-Wesley, 2012
 *
 * (C) Copyright Nicolai M. Josuttis 2012.
 * Permission to copy, use, modify, sell and distribute this software
 * is granted provided this copyright notice appears in all copies.
 * This software is provided "as is" without express or implied
 * warranty, and with no claim as to its suitability for any purpose.
 */
#include <cstddef>    // for size_t

template <typename T>
class MyAlloc {
  public:
    // type definitions
    typedef T value_type;

    // constructors
    // - nothing to do because the allocator has no state
    MyAlloc () noexcept {
    }
    template <typename U>
    MyAlloc (const MyAlloc<U>&) noexcept {
        // no state to copy
    }

    // allocate but don't initialize num elements of type T
    T* allocate (std::size_t num) {
        // allocate memory with global new
        return static_cast<T*>(::operator new(num*sizeof(T)));
    }

    // deallocate storage p of deleted elements
    void deallocate (T* p, std::size_t num) {
        // deallocate memory with global delete
        ::operator delete(p);
    }
};

// return that all specializations of this allocator are interchangeable
template <typename T1, typename T2>
bool operator== (const MyAlloc<T1>&,
                 const MyAlloc<T2>&) noexcept {
    return true;
}
template <typename T1, typename T2>
bool operator!= (const MyAlloc<T1>&,
                 const MyAlloc<T2>&) noexcept {
    return false;
}
```