```
// Austin Keith Faulkner: a_f408
// September 29, 2019
//
// FILE: sequence.template
// TEMPLATED CLASS IMPLEMENTED: sequence<Item> (see sequence.h for
//                              documentation).
// INVARIANT for the sequence class:
//   1. The number of items in the sequence is in the member variable
//      used;
//   2. The actual items of the sequence are stored in a partially
//      filled array. The array is a compile-time array whose size
//      is fixed at CAPACITY; the  member variable data references
//      the array.
//   3. For an empty sequence, we do not care what is stored in any
//      of data; for a non-empty sequence the items in the sequence
//      are stored in data[0] through data[used-1], and we don't care
//      what's in the rest of data.
//   4. The index of the current item is in the member variable
//      current_index. If there is no valid current item, then
//      current item will be set to the same number as used.
//      NOTE: Setting current_index to be the same as used to
//            indicate "no current item exists" is a good choice
//            for at least the following reasons:
//            (a) For a non-empty sequence, used is non-zero and
//                a current_index equal to used indexes an element
//                that is (just) outside the valid range. This
//                gives us a simple and useful way to indicate
//                whether the sequence has a current item or not:
//                a current_index in the valid range indicates
//                that there's a current item, and a current_index
//                outside the valid range indicates otherwise.
//            (b) The rule remains applicable for an empty sequence,
//                where used is zero: there can't be any current
//                item in an empty sequence, so we set current_index
//                to zero (= used), which is (sort of just) outside
//                the valid range (no index is valid in this case).
//            (c) It simplifies the logic for implementing the
//                advance function: when the precondition is met
//                (sequence has a current item), simply incrementing
//                the current_index takes care of fulfilling the
//                postcondition for the function for both of the two
//                possible scenarios (current item is and is not the
//                last item in the sequence).

#include <cassert>
#include "sequence.h"

namespace CS3358_FA2019_A04
{
   template <class Item>
   sequence<Item>::sequence() : used(0), current_index(0) { }
```

```
template <class Item>
void sequence<Item>::start() { current_index = 0; }

template <class Item>
void sequence<Item>::end()
   { current_index = (used > 0) ? used - 1 : 0; }

template <class Item>
void sequence<Item>::advance()
{
   assert( is_item() );
   ++current_index;
}

template <class Item>
void sequence<Item>::move_back()
{
   assert( is_item() );
   if (current_index == 0)
      current_index = used;
   else
      --current_index;
}

template <class Item>
void sequence<Item>::add(const Item& entry)
{
   assert( size() < CAPACITY );

   size_type i;

   if ( ! is_item() )
   {
      if (used > 0)
         for (i = used; i >= 1; --i)
            data[i] = data[i - 1];
      data[0] = entry;
      current_index = 0;
   }
   else
   {
      ++current_index;
      for (i = used; i > current_index; --i)
         data[i] = data[i - 1];
      data[current_index] = entry;
   }
   ++used;
}

template <class Item>
void sequence<Item>::remove_current()
{
```

```
      assert( is_item() );

      size_type i;

      for (i = current_index + 1; i < used; ++i)
         data[i - 1] = data[i];
      --used;
   }

   template <class Item>
   typename sequence<Item>::size_type sequence<Item>::size() const
      { return used; }

   template <class Item>
   bool sequence<Item>::is_item() const
      { return (current_index < used); }

   template <class Item>
   Item sequence<Item>::current() const
   {
      assert( is_item() );

      return data[current_index];
   }
}
```