```
// Austin Keith Faulkner: a_f408
// September 29, 2019
//
// FILE: sequence.h (part of the namespace CS3358_FA2019_A04)
//
/////////////////////////////////////////////////////////////////////
// NOTE: A single template class for sequence has been constructed,
//       using a single namespace.
/////////////////////////////////////////////////////////////////////
//
// TEMPLATE CLASS PROVIDED: sequence<Item>
//
// CLASS DESCRIPTION: sequence<Item> is a templated container class.
//                    The container holds a list of items. Each list
//                    may have a "current item" that is designated in
//                    the list.
//
// TYPEDEFS and MEMBER CONSTANTS for the sequence<Item> template class:
//
//   typedef ____ sequence<Item>::value_type
//     Item, the template-parameter, is the data type of the elements stored in
//     the sequence.  May also be defined as sequence<Item>::value_type.
//     It may be any of the C++ built-in types (int, char, etc.), or a class
//     with a default constructor, an assignment operator, and a copy constructor.
//
//   typedef ____ sequence<Item>::size_type
//     sequence<Item>::size_type is the data type of any variable that keeps
//     track of how many items are in a sequence.
//
//   static const size_type CAPACITY = _____
//     sequence::CAPACITY is the maximum number of items that a
//     sequence can hold.
//
// CONSTRUCTOR for the sequence class:
//   sequence()
//     Pre:  (none)
//     Post: The sequence has been initialized as an empty sequence.
//
// MODIFICATION MEMBER FUNCTIONS for the sequence class:
//   void start()
//     Pre:  (none)
//     Post: The first item on the sequence becomes the current item
//           (but if the sequence is empty, then there is no current item).
//
//   void end()
//     Pre:  (none)
//     Post: The last item on the sequence becomes the current item
//           (but if the sequence is empty, then there is no current item).
//
//   void advance()
//     Pre:  is_item() returns true.
//     Post: If the current item was the last item in the sequence, then
//           there is no longer any current item. Otherwise, the new current
//           item is the item immediately after the original current item.
//
//   void move_back()
//     Pre:  is_item() returns true.
//     Post: If the current item was the first item in the sequence, then
//           there is no longer any current item. Otherwise, the new current
//           item is the item immediately before the original current item.
//
//   void add(const Item& entry)
//     Pre:  size() < CAPACITY.
//     Post: A new copy of entry has been inserted in the sequence after
//           the current item. If there was no current item, then the new
//           entry has been inserted as new first item of the sequence. In
//           either case, the newly added item is now the current item of
//           the sequence.
//
//   void remove_current()
//     Pre:  is_item() returns true.
//     Post: The current item has been removed from the sequence, and
//           the item after this (if there is one) is now the new current
//           item. If the current item was already the last item in the
//           sequence, then there is no longer any current item.
//
// CONSTANT MEMBER FUNCTIONS for the sequence class:
//   size_type size() const
//     Pre:  (none)
//     Post: The return value is the number of items in the sequence.
//
//   bool is_item() const
//     Pre:  (none)
//     Post: A true return value indicates that there is a valid
//           "current" item that may be retrieved by activating the current
//           member function (listed below). A false return value indicates
//           that there is no valid current item.
//
//   Item current() const
//     Pre:  is_item() returns true.
//     Post: The item returned is the current item in the sequence.
//
// VALUE SEMANTICS for the sequence class:
//    Assignments and the copy constructor may be used with sequence
//    objects.

#ifndef SEQUENCE_H
#define SEQUENCE_H

#include <cstdlib>  // provides size_t

namespace CS3358_FA2019_A04
{
   template <class Item>
   class sequence
   {
   public:
      // TYPEDEFS and MEMBER CONSTANTS
      typedef size_t size_type;
      static const size_type CAPACITY = 10;
      // CONSTRUCTOR
      sequence();
      // MODIFICATION MEMBER FUNCTIONS
      void start();
      void end();
      void advance();
      void move_back();
      void add(const Item& entry);
      void remove_current();
      // CONSTANT MEMBER FUNCTIONS
      size_type size() const;
      bool is_item() const;
      Item current() const;

   private:
      Item data[CAPACITY];
      size_type used;
      size_type current_index;
   };
}

#include "sequence.template" // Include the implementation file for
                             // for templated class.
#endif
```