```
// FILE: Sequence.h
// CLASS PROVIDED: sequence (part of the namespace CS3358_FA2019)
//
// TYPEDEFS and MEMBER CONSTANTS for the sequence class:
//   typedef ____ value_type
//    sequence::value_type is the data type of the items in the
sequence.
//    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 ____ size_type
//    sequence::size_type is the data type of any variable that keeps
//    track of how many items are in a sequence.
//
//   static const size_type DEFAULT_CAPACITY = _____
//    sequence::DEFAULT_CAPACITY is the default initial capacity of a
//    sequence that is created by the default constructor.
//
// CONSTRUCTOR for the sequence class:
//   sequence(size_type initial_capacity = DEFAULT_CAPACITY)
//    Pre:  initial_capacity > 0
//    Post: The sequence has been initialized as an empty sequence.
//      The insert/attach functions will work efficiently (without
//      allocating new memory) until this capacity is reached.
//    Note: If Pre is not met, initial_capacity will be adjusted to 1.
//
// MODIFICATION MEMBER FUNCTIONS for the sequence class:
//   void resize(size_type new_capacity)
//    Pre:  new_capacity > 0
//    Post: The sequence's current capacity is changed to new_capacity
//      (but not less that the number of items already on the
sequence).
//      The insert/attach functions will work efficiently (without
//      allocating new memory) until this new capacity is reached.
//    Note: If new_capacity is less than used, it will be made equal
to
//      to used (in order to preserve existing data). Thereafter, if
Pre
//      is not met, new_capacity will be adjusted to 1.
//
//   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 advance()
//    Pre:  is_item returns true.
//    Post: If the current item was already 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 insert(const value_type& entry)
//    Pre:  none
//    Post: A new copy of entry has been inserted in the sequence
//      before the current item. If there was no current item, then
//      the new entry has been inserted at the front of the sequence.
//      In either case, the newly inserted item is now the current
item
//      of the sequence.
//
//   void attach(const value_type& entry)
//    Pre:  none
//    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 attached to the end of the sequence. In either
//      case, the newly inserted 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.
//
//   value_type 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
```

```
{
   class sequence
   {
   public:
      // TYPEDEFS and MEMBER CONSTANTS
      typedef double value_type;
      typedef std::size_t size_type;
      static const size_type DEFAULT_CAPACITY = 30;
      // CONSTRUCTORS and DESTRUCTOR
      sequence(size_type initial_capacity = DEFAULT_CAPACITY);
      sequence(const sequence& source);
      ~sequence();
      // MODIFICATION MEMBER FUNCTIONS
      void resize(size_type new_capacity);
      void start();
      void advance();
      void insert(const value_type& entry);
      void attach(const value_type& entry);
      void remove_current();
      sequence& operator=(const sequence& source);
      // CONSTANT MEMBER FUNCTIONS
      size_type size() const;
      bool is_item() const;
      value_type current() const;
   private:
      value_type* data;
      size_type used;
      size_type current_index;
      size_type capacity;
   };
}

#endif
```