```
// FILE: sequence.h
////////////////////////////////////////////////////////////////////
// NOTE: Two separate versions of sequence (one for a sequence of real
//       numbers and another for a sequence characters are specified,
//       in two separate namespaces in this header file. For both
//       versions, the same documentation applies.
////////////////////////////////////////////////////////////////////
// CLASS PROVIDED: sequence (a container class for a list of items,
//                 where each list may have a designated item called
//                 the current item)
//
// TYPEDEFS and MEMBER functions 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 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
```

```
47 //           there is no longer any current item. Otherwise, the new current
48 //           item is the item immediately before the original current item.
49 //   void add(const value_type& entry)
50 //     Pre:  size() < CAPACITY.
51 //     Post: A new copy of entry has been inserted in the sequence after
52 //           the current item. If there was no current item, then the new
53 //           entry has been inserted as new first item of the sequence. In
54 //           either case, the newly added item is now the current item of
55 //           the sequence.
56 //   void remove_current()
57 //     Pre:  is_item() returns true.
58 //     Post: The current item has been removed from the sequence, and
59 //           the item after this (if there is one) is now the new current
60 //           item. If the current item was already the last item in the
61 //           sequence, then there is no longer any current item.
62 //
63 // CONSTANT MEMBER FUNCTIONS for the sequence class:
64 //   size_type size() const
65 //     Pre:  (none)
66 //     Post: The return value is the number of items in the sequence.
67 //   bool is_item() const
68 //     Pre:  (none)
69 //     Post: A true return value indicates that there is a valid
70 //           "current" item that may be retrieved by activating the current
71 //           member function (listed below). A false return value indicates
72 //           that there is no valid current item.
73 //   value_type current() const
74 //     Pre:  is_item() returns true.
75 //     Post: The item returned is the current item in the sequence.
76 // VALUE SEMANTICS for the sequence class:
77 //    Assignments and the copy constructor may be used with sequence
78 //    objects.
79
80 #ifndef SEQUENCE_H
81 #define SEQUENCE_H
82
83 #include <cstdlib>  // provides size_t
84
85 namespace CS3358_SP2023_A04_sequenceOfNum
86 {
87    template <class value_type> class sequence
88    {
89    public:
90       // TYPEDEFS and MEMBER SP2020
91       typedef double value_type;
```

```
92        typedef size_t size_type;
93        static const size_type CAPACITY = 10;
94        // CONSTRUCTOR
95        sequence();
96        // MODIFICATION MEMBER FUNCTIONS
97        void start();
98        void end();
99        void advance();
100       void move_back();
101       void add(const value_type& entry);
102       void remove_current();
103       // CONSTANT MEMBER FUNCTIONS
104       size_type size() const;
105       bool is_item() const;
106       value_type current() const;
107
108    private:
109       value_type data[CAPACITY];
110       size_type used;
111       size_type current_index;
112    };
113 }
114
115 namespace CS3358_SP2023_A04_sequenceOfChar
116 {
117    template <class value_type>
118    class sequence
119    {
120    public:
121       // TYPEDEFS and MEMBER SP2020
122       typedef char value_type;
123       typedef size_t size_type;
124       static const size_type CAPACITY = 10;
125       // CONSTRUCTOR
126       sequence();
127       // MODIFICATION MEMBER FUNCTIONS
128       void start();
129       void end();
130       void advance();
131       void move_back();
132       void add(const value_type& entry);
133       void remove_current();
134       // CONSTANT MEMBER FUNCTIONS
135       size_type size() const;
136       bool is_item() const;
137       value_type current() const;
138
139    private:
140       value_type data[CAPACITY];
```

```
141         size_type used;
142         size_type current_index;
143     };
144 }
145 
146 #include "sequence.cpp"
147 #endif
148 
```