```
// FILE: sequenceTest.cpp
// An interactive test program for the sequence class

#include <cctype>      // provides toupper
#include <iostream>    // provides cout and cin
#include <cstdlib>     // provides EXIT_SUCCESS
#include "sequence.h"
using namespace CS3358_FA2019_A04_sequence;
using namespace std;

// PROTOTYPES for functions used by this test program:

void print_menu();
// Pre:  (none)
// Post: A menu of choices for this program is written to cout.
char get_user_command();
// Pre:  (none)
// Post: The user is prompted to enter a one character command.
//       The next character is read (skipping blanks and newline
//       characters), and this character is returned.
template <class T>
void show_list(T src);
// Pre: (none)
// Post: The items of src are printed to cout (one per line).
int get_object_num();
// Pre:  (none)
// Post: The user is prompted to enter either 1 or 2. The
//       prompt is repeated until a valid integer can be read
//       and the integer's value is either 1 or 2. The valid
//       integer read is returned. The input buffer is cleared
//       of any extra input until and including the first
//       newline character.
double get_number();
// Pre:  (none)
// Post: The user is prompted to enter a real number. The prompt
//       is repeated until a valid real number can be read. The
//       valid real number read is returned. The input buffer is
//       cleared of any extra input until and including the
//       first newline character.
char get_character();
// Pre:  (none)
// Post: The user is prompted to enter a non-whitespace character.
//       The prompt is repeated until a non-whitespace character
//       can be read. The non-whitespace character read is returned.
//       The input buffer is cleared of any extra input until and
//       including the first newline character.

int main(int argc, char *argv[])
{
   sequence<double> s1;  // A sequence of double for testing
   sequence<char> s2; // A sequence of char for testing
   int objectNum;    // A number to indicate selection of s1 or s2
   double numHold;   // Holder for a real number
   char charHold;    // Holder for a character
   char choice;      // A command character entered by the user

   cout << "An empty sequence of real numbers (s1) and\n"
        << "an empty sequence of characters (s2) have been created."
        << endl;

   do
   {
      if (argc == 1)
         print_menu();
      choice = toupper( get_user_command() );
      switch (choice)
      {
         case '!':
            objectNum = get_object_num();
```

```
70              if (objectNum == 1)
71              {
72                 s1.start();
73                 cout << "s1 started" << endl;
74              }
75              else
76              {
77                 s2.start();
78                 cout << "s2 started" << endl;
79              }
80              break;
81           case '&':
82              objectNum = get_object_num();
83              if (objectNum == 1)
84              {
85                 s1.end();
86                 cout << "s1 ended" << endl;
87              }
88              else
89              {
90                 s2.end();
91                 cout << "s2 ended" << endl;
92              }
93              break;
94           case '+':
95              objectNum = get_object_num();
96              if (objectNum == 1)
97              {
98                 if ( ! s1.is_item() )
99                    cout << "Can't advance s1." << endl;
100                else
101                {
102                   s1.advance();
103                   cout << "Advanced s1 one item."<< endl;
104                }
105             }
106             else
107             {
108                if ( ! s2.is_item() )
109                   cout << "Can't advance s2." << endl;
110                else
111                {
112                   s2.advance();
113                   cout << "Advanced s2 one item."<< endl;
114                }
115             }
116             break;
117          case '-':
118             objectNum = get_object_num();
119             if (objectNum == 1)
120             {
121                if ( ! s1.is_item() )
122                   cout << "Can't move back s1." << endl;
123                else
124                {
125                   s1.move_back();
126                   cout << "Moved s1 back one item."<< endl;
127                }
128             }
129             else
130             {
131                if ( ! s2.is_item() )
132                   cout << "Can't move back s2." << endl;
133                else
134                {
135                   s2.move_back();
136                   cout << "Moved s2 back one item."<< endl;
137                }
138             }
```

```
            break;
         case '?':
            objectNum = get_object_num();
            if (objectNum == 1)
            {
               if ( s1.is_item() )
                  cout << "s1 has a current item." << endl;
               else
                  cout << "s1 has no current item." << endl;
            }
            else
            {
               if ( s2.is_item() )
                  cout << "s2 has a current item." << endl;
               else
                  cout << "s2 has no current item." << endl;
            }
            break;
         case 'C':
            objectNum = get_object_num();
            if (objectNum == 1)
            {
               if ( s1.is_item() )
                  cout << "Current item in s1 is: "
                       << s1.current() << endl;
               else
                  cout << "s1 has no current item." << endl;
            }
            else
            {
               if ( s2.is_item() )
                  cout << "Current item in s2 is: "
                       << s2.current() << endl;
               else
                  cout << "s2 has no current item." << endl;
            }
            break;
         case 'P':
            objectNum = get_object_num();
            if (objectNum == 1)
            {
               if (s1.size() > 0)
               {
                  cout << "s1: ";
                  show_list(s1);
                  cout << endl;
               }
               else
                  cout << "s1 is empty." << endl;
            }
            else
            {
               if (s2.size() > 0)
               {
                  cout << "s2: ";
                  show_list(s2);
                  cout << endl;
               }
               else
                  cout << "s2 is empty." << endl;
            }
            break;
         case 'S':
            objectNum = get_object_num();
            if (objectNum == 1)
               cout << "Size of s1 is: " << s1.size() << endl;
            else
               cout << "Size of s2 is: " << s2.size() << endl;
            break;
```

```
208             case 'A':
209                objectNum = get_object_num();
210                if (objectNum == 1)
211                {
212                   numHold = get_number();
213                   s1.add(numHold);
214                   cout << numHold << " added to s1." << endl;
215                }
216                else
217                {
218                   charHold = get_character();
219                   s2.add(charHold);
220                   cout << charHold << " added to s2." << endl;
221                }
222                break;
223             case 'R':
224                objectNum = get_object_num();
225                if (objectNum == 1)
226                {
227                   if ( s1.is_item() )
228                   {
229                      numHold = s1.current();
230                      s1.remove_current();
231                      cout << numHold << " removed from s1." << endl;
232                   }
233                   else
234                      cout << "s1 has no current item." << endl;
235                }
236                else
237                {
238                   if ( s2.is_item() )
239                   {
240                      charHold = s2.current();
241                      s2.remove_current();
242                      cout << charHold << " removed from s2." << endl;
243                   }
244                   else
245                      cout << "s2 has no current item." << endl;
246                }
247                break;
248             case 'Q':
249                cout << "Quit option selected...bye" << endl;
250                break;
251             default:
252                cout << choice << " is invalid...try again" << endl;
253          }
254       }
255       while (choice != 'Q');
256
257       cin.ignore(999, '\n');
258       cout << "Press Enter or Return when ready...";
259       cin.get();
260       return EXIT_SUCCESS;
261    }
262
263    void print_menu()
264    {
265       cout << endl;
266       cout << "The following choices are available:\n";
267       cout << "  !  Activate the start() function\n";
268       cout << "  &  Activate the end() function\n";
269       cout << "  +  Activate the advance() function\n";
270       cout << "  -  Activate the move_back() function\n";
271       cout << "  ?  Print the result from the is_item() function\n";
272       cout << "  C  Print the result from the current() function\n";
273       cout << "  P  Print a copy of the entire sequence\n";
274       cout << "  S  Print the result from the size() function\n";
275       cout << "  A  Add a new item with the add(...) function\n";
276       cout << "  R  Activate the remove_current() function\n";
```

```
277      cout << "  Q  Quit this test program" << endl;
278   }
279
280   char get_user_command()
281   {
282      char command;
283
284      cout << "Enter choice: ";
285      cin >> command;
286
287      cout << "You entered ";
288      cout << command << endl;
289      return command;
290   }
291
292   template <class T>
293   void show_list(T src)
294   {
295      for ( src.start(); src.is_item(); src.advance() )
296         cout << src.current() << "  ";
297   }
298
299   int get_object_num()
300   {
301      int result;
302
303      cout << "Enter object # (1 = s1, 2 = s2) ";
304      cin  >> result;
305      while ( ! cin.good() )
306      {
307         cerr << "Invalid integer input..." << endl;
308         cin.clear();
309         cin.ignore(999, '\n');
310         cout << "Re-enter object # (1 = s1, 2 = s2) ";
311         cin  >> result;
312      }
313      // cin.ignore(999, '\n');
314
315      while (result != 1 && result != 2)
316      {
317         cin.ignore(999, '\n');
318         cerr << "Invalid object # (must be 1 or 2)..." << endl;
319         cout << "Re-enter object # (1 = s1, 2 = s2) ";
320         cin  >> result;
321         while ( ! cin.good() )
322         {
323            cerr << "Invalid integer input..." << endl;
324            cin.clear();
325            cin.ignore(999, '\n');
326            cout << "Re-enter object # (1 = s1, 2 = s2) ";
327            cin  >> result;
328         }
329         // cin.ignore(999, '\n');
330      }
331
332      cout << "You entered ";
333      cout << result << endl;
334      return result;
335   }
336
337   double get_number()
338   {
339      double result;
340
341      cout << "Enter a real number: ";
342      cin  >> result;
343      while ( ! cin.good() )
344      {
345         cerr << "Invalid real number input..." << endl;
```

```
346         cin.clear();
347         cin.ignore(999, '\n');
348         cout << "Re-enter a real number ";
349         cin  >> result;
350      }
351      // cin.ignore(999, '\n');
352
353      cout << "You entered ";
354      cout << result << endl;
355      return result;
356   }
357
358   char get_character()
359   {
360      char result;
361
362      cout << "Enter a non-whitespace character: ";
363      cin  >> result;
364      while ( ! cin )
365      {
366         cerr << "Invalid non-whitespace character input..." << endl;
367         cin.ignore(999, '\n');
368         cout << "Re-enter a non-whitespace character: ";
369         cin  >> result;
370      }
371      // cin.ignore(999, '\n');
372
373      cout << "You entered ";
374      cout << result << endl;
375      return result;
376   }
377
```